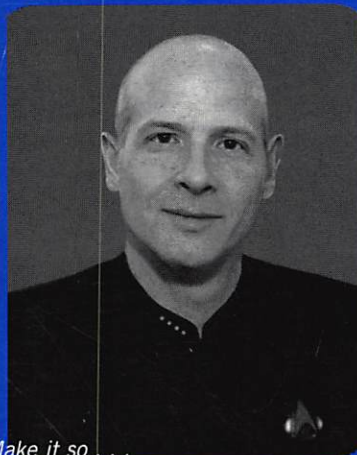


The USENIX Association Magazine

December 1998

;login:

volume 23 • number 6



Make it so . . .

inside:

CONFERENCE REPORTS

USENIX 1998 Windows NT Symposium

LISA NT Conference

USENIX Workshop on Electronic Commerce

SAGE NEWS & FEATURES

Enough SNMP to Be Dangerous

NT Accounts

Knit One, Perl 5.005

Toolman

STANDARDS REPORTS

An Overview of the POSIX 1d Draft Standard

BOOK REVIEWS

The Bookworm's Top Ten Books for 1998

Applying RCS and SCCS

Protecting Networks with SATAN

USENIX NEWS

International Programs

Twenty Years Ago in USENIX

USA Bags Gold at IOI!

features:

Source Code UNIX

by Bob Gray

Torn Money and PGP Web of Trust

by Jeanette McLeod and Greg Rose

Using Java

by Prithvi Rao

Musings

by Captain Farrow

Java I/O Performance

by Glen McCluskey

upcoming events

NordU99 - 1st Nordic EurOpen/USENIX Conference

WHEN	WHERE
February 9-12/99	Stockholm, Sweden

3rd Symposium on Operating Systems Design and Implementation

Co-sponsored by ACM SIGOPS and IEEE TCOS

WHEN	WHERE	WHO <i>program co-chairs</i>
February 22-25/99	New Orleans, LA	Margo Seltzer & Paul Leach

DEADLINES

Final Papers
January 6/99

1st Conference on Network Administration

Co-sponsored by SAGE

WHEN	WHERE	WHO <i>program co-chair</i>
April 7-9/99	Santa Clara, CA	David Williamson & Paul Ebersman

DEADLINES

Final Papers
February 23/99

1st USENIX Workshop on Intrusion Detection & Network Monitoring

WHEN	WHERE	WHO <i>program chair</i>
April 11-12/99	Santa Clara, CA	Marcus J. Ranum

DEADLINES

Final Papers
February 23/99

5th Conference on Object-Oriented Technologies and Systems (COOTS)

WHEN	WHERE	WHO <i>program chair</i>
May 3-7/99	San Diego, CA	Murthy V. Devarakonda

DEADLINES

Final Papers
March 23/99

USENIX Workshop on Smartcard Technology

WHEN	WHERE	WHO <i>program chairs</i>
May 10-11/99	Chicago, IL	Scott Guthery & Peter Honeyman

DEADLINES

Notification to Authors	Final Papers
January 12/99	March 30/99

SANS99

Co-sponsored by SAGE

WHEN	WHERE
May 9-15/99	Baltimore, MD

USENIX Annual Technical Conference

WHEN	WHERE	WHO
June 7-11/99	Monterey, CA	Avi Rubin, Program Chair Jordan Hubbard, Freenix Track Chair

DEADLINES

Notification to Authors	Final Papers
January 20/99	April 27/99

3rd USENIX Windows NT Symposium

WHEN	WHERE	WHO <i>program chair</i>
July 12-16/99	Seattle, WA	Werner Vogels & Stephen Walli

DEADLINES

Paper Submission	Notification to Authors	Final Papers
February 23/99	March 23/99	June 1/99

2nd Large Installation System Administration of Windows NT Conference (LISA-NT)

Co-sponsored by USENIX and SAGE

WHEN	WHERE	WHO <i>program co-chairs</i>
July 12-16/99	Seattle, WA	Gerald Carter and Ralph Loura

DEADLINES

Paper Submission	Notification to Authors	Final Papers
February 23/99	March 23/99	June 1/99

Eighth USENIX Security Symposium

WHEN	WHERE	WHO
August 23-26, 1999	Washington, D.C.	Win Treese, Program Chair Avi Rubin, IT Coordinator

DEADLINES

Paper Submissions	Notification to Authors	Final Papers
March 16/99	April 21/99	July 12/99

2nd Conference on Domain-Specific Languages

Sponsored by USENIX in cooperation with ACM SIGPLAN and SIGSOFT

WHEN	WHERE	WHO <i>program chair</i>
October 3-6/99	Austin, TX	Thomas Ball

DEADLINES

Paper Submissions	Notification to Authors	Final Papers
March 22/99	June 2/99	August 24/99

For a complete list of future USENIX events, access <http://www.usenix.org/events>

contents

2 IN THIS ISSUE . . .

CONFERENCE REPORTS

- 3 NTnix . . . You Are There
by George M. Jones
- 5 2nd USENIX 1998 Windows NT Symposium
- 18 Large Installation System Administration of Windows NT Conference
- 23 Third USENIX Workshop on Electronic Commerce

SAGE NEWS AND FEATURES

- 33 Coming Up to Speed
by Tina Darmohray
- 34 End of Year Report
by Hal Miller
- 36 SAGE Certification: Q&A
by Barbara L. Dijker
- 37 Are You Preparing for the Millenium?
by Tim Gassaway
- 38 Making Users Happy
by Sean Kamath
- 39 Report of the Nominating Committee
by Pat Wilson, Amy Kreiling, and Greg Rose
- 40 Enough SNMP to be Dangerous, Part 1
by Elizabeth Zwicky
- 45 NT Accounts: A Starting Point
by Phil Cox
- 47 Knit One, Perl 5.005
by Joseph N. Hall
- 50 Toolman: Generating Web Pages with sh and make
by Daniel E. Singer

FEATURES

- 56 Source Code UNIX: Help a Friend Get Online, Cheap
by Bob Gray
- 61 Torn Money and the PGP Web of Trust
by Jeanette McLeod and Greg Rose
- 67 Using Java
by Prithvi Rao
- 71 Musings
by Captain Farrow
- 74 Java I/O Performance
by Glen McCluskey

STANDARDS REPORTS

- 77 An Update on Standards Relevant to USENIX Members
by Nicholas M. Stoughton

BOOK REVIEWS

- 85 The Bookworm
by Peter H. Salus
- 86 Applying RCS and SCCS
Reviewed by Nick Christenson
- 87 Protecting Networks with SATAN
Reviewed by Nick Christenson

USENIX NEWS

- 88 USENIX International Programs
- 89 Twenty Years Ago in USENIX
by Peter H. Salus
- 89 USA Bags Gold at IOI
by Rob Kolstad

ANNOUNCEMENTS AND CALLS

- 90 3rd Symposium on Operating Systems Design and Implementation
- 92 The Fifth Annual International Conference on Mobile Computing and Networking (MobiCom '99)
- 96 *motd*
by Rob Kolstad

or target systems. His organization has been trying to emulate a "reasonable subset" of the Win32 API to demonstrate that emulation is possible under UNIX, and they have been able to implement most low-level constructs of Win32.

The problem they originally set out to solve was: Given a console application for Win32 written in Visual C++ 5.0 with use of the Standard Template Libraries (STL) and running under Windows 95 or NT, compile and execute the same code on UNIX, with gcc 2.8.1 and STL, for Solaris 2.6 (sparc/x86) or Linux 2.0 (x86).

By "a reasonable subset," Paas means implementation of a couple key areas. The first is support for NT multithreading (i.e., the ability to create, destroy, suspend, and resume preemptive threads) and for most synchronization and thread-local storage (TLS) functions. To demonstrate memory management abilities, they wanted to be able to allocate, commit, and protect virtual memory on the page level, as well as support memory mapping I/O and files. They wanted to provide user level page fault handling with structured exception handling (SEH) to emulate NT SEH. Finally (and ambitiously), they wanted to provide use of the Winsock API for TCP/IP under the emulator.

Paas then went into the implementation details of nt2unix, comparing code to accomplish common tasks such as creating a thread in NT versus POSIX versus Solaris. Creating a thread is in itself very simple, but the differences between operating systems meant they had to ignore LPSEC attributes like Windows 95 does.

One major problem with thread synchronization is that suspending and resuming threads is not possible under the POSIX thread API. Additionally, some Win32 thread concepts are hard to implement efficiently within POSIX. The NT kernel usually handles this, but in UNIX it must be done manually, which implies some performance hits.

Memory management turned out to be fairly easy. Structured exception handling wasn't as easy and couldn't be supported directly, since supporting the keywords try and except would require a change in the compiler. They decided to implement SetUnhandledExceptionFilter(), which creates a global signal handler. Mapping NT exception codes to UNIX signals, where there isn't always a good match, made this difficult.

To enable TCP/IP networking using Winsock, they decided to restrict Winsock 2.0 to the BSD Sockets API. The bulk of the task was translating data types, definitions, and error codes. Paas notes that the pitfalls in this are that some types are hard to map, like fd_set: Winsock's select() function is most definitely not BSD's select().

To test their solutions, they emulated a 15,000-line native Win32 Visual C++ code module, SVMlib. This shared virtual-memory library is all-software, user-level, and page-based. They ran this with nt2unix with no source code changes. Initial time comparisons show satisfactory behavior, the major reason for slightly slower performance than on a Win32 platform being that UNIX signal handling is significantly more expensive than Win32 event handling.

Paas's team concluded that Win32 API emulation under UNIX is very possible, and that if the emulator is application-driven, it can be implemented within finite time (three man-months). Paas says "nt2unix is a reasonable first step to develop portable low level applications." In the future they would like to implement a more complete set of Win32 base services, allowing more applications to be run under UNIX (NT services could be run as UNIX daemons, for example).

nt2unix: <<http://www.lfbs.rwth-aachen.de/~karsten/projects/nt2unix>>
SVMlib: <<http://www.lfbs.rwth-aachen.de/~karsten/projects/SVMlib>>

NT-SwiFT: Software Implemented Fault Tolerance on Windows NT

Yennun Huang, P. Emerald Chung, and Chandra Kintala, Bell Labs, Lucent Technologies; Chung-Yih Wang and De-Ron Liang, Institute of Information Science, Academia Sinica

Yennun Huang presented NT-SwiFT, a group of software components implemented to provide fault tolerance on Windows NT. These were originally developed for UNIX and have been ported to NT with new features added.

The problem is to make distributed applications highly available and fault tolerant. Huang outlined three possible solutions: (1) transaction processing as in Microsoft Transaction Server; (2) active replication / virtual synchrony as in ISIS, HORUS, and Ensemble; (3) checkpointing and rollback recovery, which is the approach SwiFT takes.

SwiFT supports three types of process replication for rollback recovery: cold, warm, and hot. Cold is fail-over with or without checkpointing. Warm is primary backup with state transfer. Hot uses an active process group with no shared memory. Regardless, the overall philosophy was to keep the error recovery mechanism transparent from client programs, and to enhance fault-tolerant server programs with fault tolerant APIs.

After an extremely detailed catalog of the many components of SwiFT and what they can be used for, Huang provided some general information about SwiFT in general. UNIX SwiFT has been used in Bell Labs for more than five years and is used in more than 20 products and services. Its technologies have been licensed to a few companies. A few projects in Lucent are trying NT-SwiFT.

SwiFT was originally ported to NT on UWIN but was re-implemented with many new features. UWIN 1.33 works, but not quite. By writing new driver code, they obtain less software dependen-

cy, a new GUI, new features, and a much needed understanding of NT internals.

The basic procedure is that when a process is initially created, important system calls and events are intercepted and recorded. This is a sneaky and very transparent solution: a whole process space can be set up (using NT calls `VirtualQuery()` and `VirtualProtect()`). Handles can be restored with library injection techniques and modification of import address tables. For client-server applications, they use an intermediate NDIS driver, which allows them to set up a dispatcher and server node with the same IP address. The dispatcher node can be failed over.

Huang gave a very interesting demo of the checkpoint and process-space recovery features using the beloved minesweeper application. This was both humorous and very effective: after placing a few mines, he took a checkpoint, placed a few more, and, naturally, "blew up" when he clicked a bad square. Then he restored from the last checkpoint, which, he explained, actually launched the process, restored its process space, and replayed a sequence of events. This allowed him to "try again" and checkpoint again when he clicked a few good squares. While trivial, it clearly demonstrated the possibilities of the system.

Huang expressed a few opinions about NT, namely that it has too many APIs and libraries (really?), but that it is very powerful and that "everything is possible in NT." It has many useful facilities, and although the OS can be an esoteric maze at times, a good point is that if you have a problem, someone somewhere has probably written some code to solve it, and it's fairly easy to find free code samples.

Huang stated a few future goals for SwiFT. They'd like to bring it to Windows 98. They are planning to add a few components (CosMic, adrejuv), more NT system calls trapping, and more dispatching algorithms for ONE-IP. They'd also

like to see SwiFT for distributed objects (CORBA, DCOM, and JAVA). Finally, they'd like to integrate SwiFT with other tools (MSCS) and NT5.

In the Q/A session, someone wanted to know about availability. The answer wasn't very clear, but it's under license and at this point is not very available (still in progress). People had many questions about the Winmine demo. Huang made it clear that GDI objects like brushes can not be captured: there's no way to understand them outside of a process space. For the demo, they use window handles only. Someone wanted to know if it was possible to save on one machine and recover on another, as this would be a very useful feature for load balancing. The answer is yes, but it has to be exactly the same type machine with the same configuration because of memory internals. Someone wanted to know if this would be able to run more than one process per server (for example, could you run thousands of SwiFT-backed objects on a server?), and could SwiFT run for days without crashing? The answer: "We're working on it."

Session: Threads

Summary by Kevin Chipalowsky

A Thread Performance Comparison: Windows NT and Solaris on a Symmetric Multiprocessor

Fabian Zabatta and Kevin Ying, Brooklyn College and CUNY Graduate School

Kevin Ying began by observing that the cost of multiprocessing equipment has dropped drastically over the past few years. A dual-processor IBM SP2 sold for \$130,000 in 1995, and a more powerful system built with Pentium II processors costs around \$13,000 today. With this much computing power readily available, mainstream operating systems need to support multithreading.

Both Windows NT and Solaris support kernel-level and user-level subprocess

objects of execution. Windows NT calls its kernel objects "threads" and its user objects "fibers." The application programmer has complete control over the scheduling of fibers. Solaris calls its kernel objects "Light Weight Processes" (LWP) and its user objects "threads." Unlike NT, Solaris provides a user level library to schedule threads to run on LWPs.

In their research, Zabatta and Ying performed seven experiments to test the relative multiprocessing performance of the two operating systems. They tested kernel execution objects in Windows NT only, but tested bound, unbound, and restricted concurrency level threading in Solaris. A bound thread in Solaris's user level library is a single thread that is always scheduled on a single LWP. An unbound thread is dynamically scheduled on a dynamically chosen number of LWPs, but the concurrency level can be restricted to a fixed number. In their concurrency restricted case, Zabatta and Ying limited the library to four LWPs (CL=4), because their experimental system had four processors.

Ying explained that neither operating system documented a limit on the number of kernel execution objects it could create. Their first experiment was to discover this limit. They found the Windows NT limit to be around 9800, and the Solaris limit to be around 2200. The second experiment tested normal thread creation speed. They wrote a simple program to create threads in a loop. Performance of NT and Solaris bound threads were very comparable. However, unbound Solaris threads could be created much faster. They also tested thread creation speed while the system was under a heavy load. In this situation, the creation of all types of Solaris threads was drastically faster than creation of Windows NT threads. Ying informed us that this could be expected, because NT gives a higher priority to threads that have been running for a long time, while Solaris gives a higher priority to newly created threads.

In the fourth experiment, performance was measured for an application requiring no synchronization. They found no major differences in running times by any of the threading models. This led them to conclude that the Solaris threading library does not significantly affect performance. Next, they tested performance in an application making heavy use of synchronization. Windows NT has two different types of synchronization objects. A critical section has local scope, and a mutex has global scope. Solaris only has critical sections, but it has a creation flag, which determines scope. Zabatta and Ying found that Windows NT critical sections drastically outperform local Solaris ones. However, global synchronization objects in Solaris outperform the global ones in NT. In the sixth experiment, they tested performance using the classic symmetric traveling salesman problem. The significant result was an almost linear speedup with parallelism. All threading models had very similar performance. The final experiment attempted to mimic real world applications with CPU bursts. They tested each threading model with drastic CPU bursts and found the restricted concurrency Solaris threads slightly outperform the others. Ying attributed this to Solaris' two-tier system.

Ying concluded by reiterating the scalability of each model, the flexibility of Solaris's design, and the performance advantages of NT's critical sections.

A System for Structured High-Performance Multithreaded Programming in Windows NT

John Thornley, K. Mani Chandy, and Hiroshi Ishii, California Institute of Technology

John Thornley opened by reminding us of a time-honored idea: multithread programs on multiprocessor computers to make them run faster. However, the idea of multithreaded programming has still had very little impact on mainstream

computing. Thornley asks and tries to explain why this is so.

In his explanation, there are three types of obstacles to the widespread adoption of multithreaded systems: the availability of symmetric multiprocessing (SMP) computers, the lack of programming systems, and the difficulty of software development. Until recently, SMP technology was been very expensive and rare. Software tools were always limited. Most were primitive and the product of academic research. They have always been unreliable, nonportable, and difficult to program. Things are changing. SMP computers are finally becoming cheap enough for their use to spread beyond expensive research labs. Commodity operating systems, notably Windows NT, support threaded software. Multithreaded programming, however, remains difficult. This is the focus of the research Thornley presented.

Why is programming so tough? Thornley argues that the problem is a lack of structure. Current tools are designed for systems programming, which is small subset of all programming that could benefit from SMP computers. Current synchronization operations are also very error-prone because of their nondeterminism. These tools are at the level of "goto" sequential programming. We need structured design techniques, modeled after tried-and-true sequential techniques. We need direct control of threads and synchronization. We need determinacy unless explicit nondeterminacy is required. And we need performance that is portable across different hardware and with different background loads.

The authors developed Sthreads, a new package of tools to deliver this functionality. The programming model is "multithreaded program = sequential program + pragmas + library calls." They claim that if a programmer follows the rules of the model, multithreaded execution is equivalent to sequential execution. This determinacy has many important conse-

quences simplifying software development. For example, a program designed for multithreading can be run sequentially for debugging.

Sthreads is not a parallelizing compiler. Their pragmas are not hints; they are specific directives. They are used around blocks and "for" loops and indicate the section of code that should be explicitly multithreaded. The Sthreads library provides counters to guarantee correct order of execution, but also provides access to traditional locks for nondeterministic programming.

Thornley presented a trivial code example that multiplies matrices. A pragma indicating that it should be multithreaded precedes the outer "for" loop. His second example was a little more complicated. It sums up arrays of floating point numbers. Since floating point arithmetic is not associative, the order of execution matters. To ensure equivalency to sequential execution, Thornley's example uses a counter. It guarantees sequential ordering and mutual exclusion in a section of code. The use of Sthreads for this example is far simpler than using the Win32 thread API.

The researchers theorize that this is all you need to make programs run fast. They think that hardware and operating system software are ready for multithreading of commodity software. Thornley made the very strong statement that if multithreaded programming is not this simple, then it will never become mainstream. He ended with the following testimonial. They took a difficult aircraft route optimization problem and implemented a solution using the Sthreads tools. In the end, their solution running on an SMP system with four Intel processors outperformed a Cray supercomputer solving the same problem using an implementation designed with traditional programming techniques.

A Transparent Checkpoint Facility On NT

Johnny Srouji, Paul Schuster, Maury Bach, and Yulik Kuzmin, Intel Corporation

Paul Schuster and Johnny Srouji presented their research and resulting checkpoint tool. Checkpointing is the act of capturing the complete state of a running process. Once captured, it can later be used to resume the process either on the same machine or be migrated to another.

In the past, many checkpointing tools have been built for UNIX systems, but this group began the development of their tool not knowing of any others for Windows NT. Given NT's increased usage over the past few years, they believed that such a tool was definitely needed.

The motivation for checkpointing is strong. It is a good way to prevent the loss of data that is due to the failure of a long-running process. It can also be used for debugging, to determine why that long-running process failed and resulted in data loss. Most significantly, checkpointing can be used to migrate a process from one machine to another in a distributed environment, possibly to improve resource utilization.

There were a number of design goals in the project. Foremost, it needed to be transparent to the running application, so no source code changes could be required. Obviously, it needed to be correct, but with a minimal performance impact. It also had to be application-independent and support multithreaded processes. The designers tried to make their implementation as portable as possible, although they discussed only the NT implementation.

For a checkpoint facility to be correct, it needs to capture the complete state of a process. Schuster and Srouji illustrated a layering of process state components. User objects, such as memory and thread contexts, were on the bottom. System state objects were above those, and GUI and external state objects were at the very

top. Moving up the layers, the complexity of capturing state increased. Schuster and Srouji said they did not even attempt to capture state for the highest layers. Their tool is limited to console applications.

It has both a user interface and a developer interface. The user runs an application using an alternate loader, which configures the app to run with automatic checkpointing. Alternatively, the application developer can explicitly control when checkpointing occurs by using a provided API.

Schuster and Srouji described the architecture of the checkpoint tool. Their checkpoint DLL is loaded into the user memory space by the loader. Its DllMain is called first, which rewrites the Import Address Table (IAT). In doing so, it forces all Win32 API calls to be redirected to checkpoint DLL functions.

When it is time to perform a checkpoint, the tool has access to all needed states. User state is in user memory, and since the tool is implemented as a DLL running in user memory space, it can directly access it. State associated with system calls is stored in system memory. The checkpoint DLL does not have access to that memory, but it can infer the system's internal state because it had a chance to see all system calls.

To resume a process, the checkpoint tool loads the application suspended. It rebuilds the state in the reverse order it captured it. Finally, it releases the application threads and they runs as if they were never stopped.

As implemented, Schuster and Srouji's work has a few limitations. Most relate to external state, which they cannot control. If a process creates any temporary files, they must still exist in their checkpointed states during resume. Any applications that bypass the IAT (by using GetProcAddress, for example) might not resume correctly. And their tool does not even attempt to deal with simultaneously checkpointing multiple processes that

interact. In the future, they plan to continue their research with more optimizations and more comprehensive API support. They hope to improve performance with incremental memory dumps. Eventually, they hope to use their checkpointing tool for process migration.

Poster and Demo Session

Summary by Michael Panitz

The Poster and Demo session was a popular, well-attended event featuring many research projects in a wide range of areas, from dynamic optimization to process migration to NT-UNIX connectivity. The session began with the session chair, John Bennett, inviting project presenters to give a one-minute summary of their projects. After the summaries, the audience was free to roam about, conversing both with the project presenters and among themselves.

Many of the projects focused on network-related advances. A Bell Labs/Microsoft team collaborated on a project that exploited COM's custom marshalling ability to run DCOM on RMTP, a multicast network protocol. A group from Harvard presented a cluster-based Web server in which page requests are preferentially forwarded to specific nodes, thus significantly increasing performance by decreasing the total number of pages each node is expected to serve. The Milan/Chime project demonstrated a distributed shared memory system which was used to support distributed preemptive scheduling (and task migration). Martin Schulz, from the SMiLE group at TU-Munich, presented a system for building a shared memory system, which supports transparent, cluster-wide memory by exploiting SCI's hardware DSM support. Finally, the Brazos parallel programming environment was presented, which facilitates parallel programming by offering features such as being able to run a Brazos program on uniprocessor, SMP, or clustered computers without recompilation.

Two posters dealt with connecting NT to other systems. Motohiro Kanda presented a mainframe file system browser, which allows one to access to file system of a Hitachi 7700 mainframe from Windows NT. Network Appliance demonstrated a specialized, multiprotocol file server that utilizes the SecureShare technology that was described in the Mixing UNIX and NT technical session. The main feature of the NetApp file server was that it supports both UNIX and NT file sharing. It allows NT clients to access UNIX files and vice versa, in addition to NT to NT/UNIX to UNIX access, all without client modification.

In a class by itself was SWiFT (not to be confused with the checkpointing project NT-Swift), a toolkit to build adaptive systems. The system is based on feedback control theory, and seeks to apply hardware control theory to software problems. In doing so, it facilitates the use of modular control components, explicitly specified performance parameters, and from there, the automatic, dynamic reconfiguration of software modules for good performance despite a changing environment.

KEYNOTE ADDRESS

Buying Computing and Storage by the Slice

Gary Campbell, Tandem Computers Inc.

Summary by Dan Mihai Dumitriu

Gary Campbell gave a compelling argument for cluster technology as a more scalable and cost-effective replacement for SMPs, MPPs, and Supercomputers. He argues that in today's computing world, SAN (System Area Network) technology has matured to the point where it is a feasible interconnect for clusters.

Key technologies that must exist in order for "computing by the slice" to succeed are: x86 SMP systems, which are very inexpensive today; balanced PCI; SAN interconnects such as VI (Virtual

Interface)-based solutions; and parallel programming standards.

Alternative technologies to clustering are SMPs, which do not scale indefinitely and do not have the best price-performance curve, and CC-nUMA (Cache Coherent Non-Uniform Memory Access). When applications start to get broken up on a nUMA machine, it starts to look more and more like a cluster. In addition, both of these technologies have single points of failure, whereas clusters are architecturally ready for fault-tolerant features.

The hardware necessary for computing by the slice is available, but the software side still needs work. "Legacy" cluster systems – such as Tandem NSK, IBM SP2, and Digital UNIX – are too difficult to replicate. More recent products such as Microsoft's cluster service, affectionately (?) called "Wolf Pair," do not scale well. Much work is needed in the software and the distributed APIs.

Some case studies of "computing by the slice": The IBM DB2 database running on 2 P6-200MHz machines interconnected with ServerNet gets 91% scaling. The Inktomi Web search engine, which is a Berkeley NOW derivative, is built out of 150 dual-processor UltraSparc II machines connected with Myrinet. This highly parallel search engine can index 110 million documents and is highly economical. The Sandia Allegra Model was originally built on Cray, later on Paragon. Now it is running on DEC Alpha's interconnected with Myrinet.

The conclusion was that computing by the slice offers superior price/performance, is architecturally ready, has lower time and cost to market, and is even gaining ground in traditionally supercomputer applications like sparse matrix computations, and also in commercial parallel databases. The architecture is primarily limited by the software. In the future look for COM+, Java EJB, and other perhaps more dramatic developments.

KEYNOTE ADDRESS

Here Comes nUMA: The Revolution in Computer Architecture and its OS Implications

Forrest Basket, Silicon Graphics Inc.

Summary by Dan Mihai Dumitriu

Forrest Basket presented an argument for CC-nUMA (Cache Coherent Non-Uniform Memory Access). He asserted that the nUMA architecture is inevitable in today's computing world, and he presented a successful implementation of the hardware and software.

Basket pointed out some problems that arise in modern systems: faster buses must be shorter and run hotter, and faster CPUs run hotter and so need more volume for cooling. Using point-to-point wires rather than buses allows you to run them faster and cooler.

The SGI Origin 2000 is a successful implementation of CC-nUMA. It has an integrated nUMA crossbar and a fat-hypercube interconnect. It has a coherence and locality protocol, 64-bit PCI, and some fault-tolerant features. Each node in the Origin 2000 has two MIPS R10000 CPUs.

The operating system is Irix 6.5. The computational and IO semantics of the system are the same as for an SMP. For a small nUMA system an SMP OS will work, as will SMP applications. Some disadvantages are additional levels in memory and IO hierarchy. Even though the system has a high-performance interconnect, latency in memory access is an issue, as is the contention for resources between nodes.

According to Basket, in order to optimize performance of parallel applications, we want to be able to specify the topology of the system as well as affinities for devices, and to be able to do this without modifying binaries. Other issues that arise are page migration between nodes and memory placement policies. Modifications to the OS kernel are necessary to support

being able to specify the initial placement of applications consistent with the user-specified system topology, replication of the kernel at boot time, a reverse page table, and a page locking scheme. System management is also an issue with nUMAs. The ability to partition systems so we can perform administrative shutdown of parts of the system is desirable, as is a sophisticated batch system that would enable users to see consistent running times.

SMP and DSM (Distributed Shared Memory) (cc-nUMA is a variant of DSM) are displacing vector supercomputers and MPPs.

Session: Mixing UNIX and NT

Summary by Michael Panitz

Merging NT and UNIX Filesystem Permissions

Dave Hitz, Bridget Allison, Andrea Borr, Rob Hawley, and Mark Muhlestein, Network Appliance

Dave Hitz presented a fast and witty overview of the WAFL file system, which enables network-based file sharing with both UNIX and NT clients. Network Appliance has created a specialized file-sharing device that uses WAFL to ease file sharing in a mixed NT/UNIX environment. The three design goals of WAFL are: to make WinNT/95 users happy by providing a security model that mimics NTFS; to keep UNIX users happy by providing a security model that mimics NFS; and to allow Windows and UNIX users to share files with each other.

Difficulties arise because UNIX (and its Network File System, NFS) and NT (and its Common Internet File System, CIFS) are fundamentally different, both in security models and in such aspects as case sensitivity (NTFS is case-insensitive, NFS is case-sensitive). NFS uses divides permissions into (user, group, world), while CIFS uses Access Control Lists (ACLs). CIFS uses a connection-based authentication scheme, while NFS is stateless. WAFL

was primarily designed to bridge these two filesystems in the most secure manner possible, while secondarily providing as intuitive an interaction as possible.

In addition to moderating access to files based on permissions, a filesystem is expected to display permissions, to allow them to modify these permissions when appropriate, and to be able to specify the default permissions to assign to a newly created file. WAFL uses both permission mapping and user mapping to accomplish these goals. When a UNIX client accesses an NT file, access is determined by UNIX-style permissions that are generated from the ACL via a process called "permission mapping." These "faked-up" permissions are guaranteed to be at least as restrictive as the NT ACL. When an NT client requests access to a UNIX file, access is determined by mapping the NT user to a UNIX account, via a process called "user mapping." The presentation argued that this was an effective, direct way to allow access in a secure, reasonably intuitive manner.

The presentation finished by touching on some of the issues surrounding WAFL, such as how to store NT ACLs, and on the administrative protocols used by NT.

Pluggable Authentication Modules for Windows NT

Naomaru Itoi and Peter Honeyman, University of Michigan

Naomaru Itoi began the presentation with an anecdote about the motivation for creating a Pluggable Authentication Module (PAM) on NT. At the University of Michigan there existed authentication modules for both Kerberos and NetWare. This was great, but the authors wanted a module that provided authentication for both Kerberos and NetWare together; the only way to accomplish this was to create a third module. To create this under NT would have been difficult and time-consuming. They wanted an authentication system would allow the user to log on once, yet use many services ("single sign-

on"), a system that would be easy to administer, and a system that would be relatively easy to develop new authentication modules for. What was wanted was a dynamic security system for NT, much like the PAM system that provides dynamic security for Linux and Solaris.

After explaining why such a system would be useful, the speaker gave an overview of PAM, which is a de facto standard for administration, being part of Linux, Solaris, and the Common Desktop Environment (CDE), and also being standardized by the IETF. PAM allows for security (re)configuration via a simple text file, which allows the administrator to specify such things as which services (Kerberos, NetWare, etc.) are required for, say, a logon attempt, or ftp session; which are optional; which services should be logged on to using the login password the user provides; which should be logged in to using a password stored in a password file, etc. PAM also provides a high-level API for authentication, so that different services can be wrapped and then configured without a recompilation.

Itoi outlined a plan of attack by next explaining GINA, the administrator-replaceable "Graphical Identification and Authentication" user authentication component. GINA enables the administrator to replace the default user authentication module with another, but still suffers from the problem of having to write one module for the Kerberos service, one for the NetWare service, and a third for Kerberos and NetWare. Further, each module would have to be configured in its own way, thus making administration of any significant number of NT machines nearly impossible. Last, custom GINA modules require special debugging tools and the use of difficult techniques, since GINA is run before anyone logs in. The plan was to build a custom GINA that implemented a subset of PAM, so that NT could be used, administered, and developed for as easily as the UNIX PAM systems.

The design and implementation of PAM, named NI_PAM, was presented, including the API supported by NI_PAM; a diagram that showed which DLLs replaced the GINA.dll and their interaction was explained.

Itoi reported that much of PAM has been successfully implemented, though more features need to be implemented, and more testing needs to be done, before a large-scale rollout can take place. The presentation concluded with some thoughts on alternate means of implementing PAM on NT, and possible future directions of the work, such as use of smartcards and screen saver locks.

Montage - An ActiveX Container for Dynamic Interfaces

Gordon Woodhull and Steven C. North, AT&T Laboratories - Research

Montage grew out of an effort to create a Windows-friendly port of an abstract graph/network editor from UNIX. The Windows graph editor would integrate with Windows applications using ActiveX (also known as OLE - Object Linking and Embedding), a runtime, object-oriented technology. The edges and nodes of the graph would be embedded objects, and the application itself would be an embeddable ActiveX container, which sounded easy enough.

Unlike previously available containers, Montage separates both the layout and control of the contained objects and the interface used to control them from the container. Thus, Montage is actually an externally controllable object container that is being used to create a graph application, Dynagraph. Montage is itself an embeddable, customizable ActiveX object, and allows dynamic changes to the layout of contained objects. Thus, Montage could be used to display the current state of a computer network, unlike something like dotty, which is used to generate static graphs. All policy decisions (i.e., which objects should be placed where, what size should they be,

etc.) are implemented objects independent of the Montage objects, thus allowing one to change the style of layout without recompiling Montage (unlike an VB MFC application).

Montage exploits the OCX96 technology of "transparent controls" to provide for different modes of interaction with the objects. This allows Montage to provide a "Viewing Mode," in which the user can view but not change the graph, and an "Editing Mode," in which the user can both view and edit the graph. At the same time, the contained objects themselves are allowed to request that their properties be set to a certain value. A contained object could, for example, request to be moved to point (x, y), and its foreground color set to blue, or to be brought to the front. Montage then forwards this request to the layout control engine, which then has the option of ignoring the request or interpreting it if it so chooses. Montage exploits the new technology of "windowless controls" to provide for different modes of interaction with the objects. This allows Montage to provide a "Viewing Mode" in which the user can view but not change the graph, and an "Editing Mode," in which the user can both view and edit the graph.

The presentation included an impressive live demo, which showed embedding a Word snippet into Montage, and then embedding a Montage graph into Word.

Session: Networking and Distributed Systems

Summary by Hui Qin Luo

SecureShare: Safe UNIX/Windows File Sharing through Multiprotocol Locking

Andrea J. Borr, Network Appliance, Inc.

Dennis Chapman, who made the presentation for author Andrea Borr, employed illustrative examples to demonstrate the capabilities of SecureShare. SecureShare is Network Appliance's solution to multiprotocol file sharing between two differ-

ent file systems, UNIX's Network File System (NFS) and the Windows Common Internet File System (CIFS) or "(PC)NFS."

SecureShare is a Multiprotocol Lock Manager providing file-sharing capabilities between UNIX clients using NFS and Windows clients using CIFS without violating data integrity. CIFS has hierarchical locking and mandatory locking functionality that requires file-open and lock retrieval before performing any operations such as reading, writing, or byte range locks. Unlike CIFS, UNIX's NFS has a nonhierarchical, file-open deficient and advisory locking mechanism. Its has no predeclarative functionality that specifies the kind of access mode it needs to a file. Therefore, these differences make file sharing between the mixed network environment difficult, if not impossible. SecureShare's main selling point is the preservation of multiprotocol data integrity by reconciling the locking mechanisms and file-open semantics between the two different file systems, and multiprotocol oplock ("opportunistic locks") management involving oplock requests from NFS to CIFS oplock break.

CIFS opportunistic locks (with the exception of level II oplocks) represent the equivalent of a file open with Read-Write/Deny-All access mode. However, access attempts by other clients (using either CIFS or NFS) to the oplocked file can cause the server to revoke the oplock through an oplock break protocol. The client who obtained the oplock gains the advantages of read-ahead on the open file, cache write operations to files, and cache lock requests. In this way, the network traffic to the file server is minimized. Chapman discussed the oplock break protocol in a mixed CIFS and NFS environment. When another client wishes to access the file, the client's request is suspended. Afterwards, an oplock break message is sent to the operating system of the CIFS client holding the oplock. The client operating system can close the file and pipe all the changes of the file stored

in the cache to the file server. It can also pipe all the cached changes and remove all the read-ahead data. It then transmits a reply to the fileserver acknowledging the break.

One of the concerns brought up in the Q/A session was the handling of a situation in which a client fails to respond to the oplock break request sent by the server due to attempted access to the oplocked file by NFS. Chapman claimed that there is an automatic session timeout on the oplock held by the client's operating system, which, if triggered, automatically relinquishes the stale batch oplock.

Session: Networking and Distributed Systems

Summary by: Hui Qin Luo

Harnessing User-Level Networking Architectures for Distributed Object Computing over High Speed Networks

Rajesh S. Madukkarumukumana, Intel Corp.; Calton Pu, Oregon Graduate Institute of Science and Technology; Hemal V. Shah, Intel Corp.

The introduction of high-performance user-level networking architectures such as Virtual Interface (VI) lays the ground work for improving the performance of distributed object systems. This presentation by Rajesh Madukkarumukumana examined the potential of custom object marshalling using VI, along with issues involved in the overall integration of user-level networking into high-level applications.

Component-based software like Distributed Component Object Model (DCOM) uses remote procedure call (RPC) mechanisms to facilitate distributed computing. Although distributed computing has matured over time, the protocols that are relied on to transport data have remained virtually unchanged, hindering the overall performance of networks such as SANs (System Area Networks). The low-latency of user-level

architectures provides an attractive solution to the problem in SAN environments. Madukkarumukumana chose to use DCOM and VI as the subjects of his research. He presents his methodology in integrating the VI based transport and a preliminary analysis of the performance results.

The VI architecture provides the illusion that each process owns the network; many performance bottlenecks are bypassed, including the operating system, to achieve this low latency, high bandwidth connection. At the heart of the standard lies two queues for each process, one for sending data, the other for receiving it; the queues contain descriptors that state the work that needs to be done. Prior to data transfer operations, a process called memory registration is performed, allowing the user process to attach physical addresses to virtual ones. Unique memory regions are referenced by these address pairs, eliminating any further bookkeeping. Two data transfer operations are accounted for – the standard send/receive operations, and Remote DMA (RDMA) read/write operations.

DCOM is a network version of COM, used for the development of component software. The network extensions in DCOM allow for all objects to be addressed the same way, hiding their location. Encoding and decoding data for transfer is called marshalling and unmarshalling, respectively; the process of marshalling and unmarshalling creates a stub object in the server process, and a proxy object in the client process. Basically, three types of marshalling are used, but the one that Madukkarumukumana discussed is custom marshalling: it allows for the object to dynamically choose how its interface pointers are marshalled.

In order for VI to do its job, the interface that DCOM uses to generate the stub and proxy, referred to as IMarshal, has to be exposed. Specialization in the object implementation is used to expose the

IMarshal interface. By exposing the parameters of the IMarshal interface, new methods can be written to make use of the VI send and receive queues. Information can therefore be sent using the VI standard, instead of the old UDP protocol. Since VI guarantees a certain quality of the signal transferred over a line, much of the overhead and interrupts involved in UDP is eliminated.

In discussing the results of his research, Madukkarumukumana stated that latency in the signal (for one-way transfer) dropped by about 30% to 60% in some cases, even only under VI emulation. The existence of core VI hardware provides a further dramatic increase in performance, and an even greater performance boost may be expected if new procedures catering to distributed computing systems are implemented within VI (results forthcoming).

Implementing IPv6 for Windows NT

Richard P. Draves, Microsoft Research; Allison Mankin, University of Southern California; Brain D. Zill, Microsoft Research

This presentation focused on the implementation and design details of IPv6 for Windows NT as well as the common pitfalls/challenges encountered in the process. IPv6 is the next generation Internet Protocol (IP) worked by the IETF. The major driver behind it and some of its key features were briefly mentioned in the paper; however, anyone new to IPv6 who wishes to know more about its history and the IPv6 specification should consult the relevant RFC documents referenced in the paper.

The presentation started with an excellent overview of the Windows NT networking architecture and how the IPv6 protocol stack can be/is integrated into it. This was followed by a high-level overview of the presenters' IPv6 implementation and some discussions of four challenges/issues encountered and the specific solution used. The presentation

ended with some notes on the implementation's performance.

The segment on NT networking internals was very informative, especially for novices. The details on the interfaces (documented or undocumented) and protocols presented, along with the helpful references mentioned in the paper, will prove useful for anyone trying to implement a different network protocol stack for NT and even for Windows 95 (due to similar network architectures). The integration of IPv6 into this networking architecture was also highlighted. Mainly, a Winsock DLL module was added to provide user-level socket functionality for IPv6 addresses, and a new TCPIP protocol driver to replace the IPv4. The implementation was "single stack," supporting only IPv6, and though not efficient was useful in isolating problems in the IPv6 stack during testing. The logical layout of the implementation was divided into three layers similar to IPv4 – the link layer, the network layer (IP), and the transport or upper layer which includes protocols such as TCP, UDP, and ICMP.

Four noteworthy problems and their implemented solutions were discussed. They range from inefficiencies in lower-level network device handlers during a receive cycle to deadlock avoidance issues.

Performance measurements for the implementation were taken using TCP throughput as the indicator metric and compared to the IPv4 stack. Results presented showed marginal performance degradation with the IPv6 stack (2.5% over 10Mb/s LAN), and somewhat higher than expectations (1.4% based on increased IPv6 header length). This is to be expected, as the developers never intended this to be an optimized implementation, but rather as a base for further research and to give Microsoft the push for a product release in the future. Whether we will see better performance results in Microsoft's official product

implementation of IPv6 is still to be determined. Comparative performance measurements against other IPv6 implementations (Solaris, Digital Unix, BSD variants etc.) were left out. The metric of comparing each implementation's relative performance to its IPv4 counterparts can be used as an indicator. Direct IPv6 TCP throughput comparisons might not be fruitful because of differences in the O/S architecture each implementation was targeted for, unless IPv4 performance was similar across these platforms. Source code size comparisons were done against another publicly available IPv6 implementation (INRIA IPv6).

"Great sample code" is available at <http://research.microsoft.com/msripv6> for anyone who wishes to dabble in Windows NT network protocol development or have a starting code base for further IPv6 research and experimentation. A more full-fledged release with security, authentication, and mobility support is expected to be available in the future.

Session: Real-time Scheduling

Summary by Jason Pettiss

A Soft Real-time Scheduling Server on Windows NT

Chih-han Lin, Hao-hua Chu, and Klara Nahrstedt, University of Illinois

Hao-hua Chu spoke about his group's implementation of a software realtime CPU scheduler for Windows NT. NT schedules applications indiscriminately based on multi-user time-sharing. Multimedia performs poorly under these conditions, especially if non-time-sensitive-conscious but CPU-hungry tasks like compilation are occurring in the background. The scheduling server is a daemon from which applications can request and acquire periodic processing time.

The scheduler requires no kernel modifications, uses the rate monotonic algorithm, supports multiple processors (SMP model), and provides guarantees for timeshare processes so that they aren't

starved by realtime tasks. Chu says there is "reasonable" performance at this point, the main problem being limited overrun protection due to the fact that the scheduler itself is a process and sometimes isn't woken up on time by NT.

The architecture consists of a broker, which handles reservation requests, builds a dispatch table, and fills the available slots of the table. The dispatcher is in charge of reading the table and responding appropriately. The dispatch table is configurable for the number of processors, the number of available slots, and the time-slice of slots. Dispatching occurs by changing the thread and process priority of participating applications between idle and highest priority realtime (1-31).

To test, Chu's team used a dual Pentium 200 with 96 MB RAM (an HP Vectra XU). Time-slice was set to 20ms. They ran two processes running MPEG decoders at FPS, one Visual C++ compilation of the MPEG decoder, and four processes running computations of sine and cosine tables. Dispatch latency worked out to be about 640 microseconds, which was longer than they would have liked, but not too large to disrupt scheduling. Performance of the two time-sensitive processes was improved, Chu noted.

The main problem, reiterated Chu, was that NT sometimes did not wake up the dispatcher on time. Also, the dispatcher, being an NT process itself, cannot preempt real-time processes. This means there is weak overrun protection. The provided NT timers weren't accurate enough, so they are using a Realtime Extension (RTX) by Venturcom to get under 1ms resolution.

They have much future work planned. Support is planned for varying processing time per period and for a process service class, similar to ATM traffic classes. They hope to run conformance tests. Also, they would like to adapt a multimedia decoder to increase reservation or decrease quality

as necessary. An additional feature, probing and profiling, could be added to figure out how much processing time to reserve on a per-application basis.

Vassal: Loadable Scheduler Support for Multi-Policy Scheduling

George M. Candea, Oracle Corporation; Michael B. Jones, Microsoft Research

George Candea presented Vassal, a system that utilizes loadable schedulers to enable multi-policy, application-specific scheduling on Windows NT. He led off with the example of a speaker late for a presentation (an application) that knows where it needs to be and when, and a cab driver (the OS) that can get him there on time if he can communicate with him. Windows NT is more like a cab driver who hasn't learned English yet – the operating system multiplexes the CPU among tasks, unaware of their individual scheduling needs.

Since no single algorithm is good enough for all task mixes, explained Candea, a compromise would be to hardcode more than one scheduling policy into the kernel. But even better would be a dynamically extensible set of policies, made possible by separating policy (scheduling) from mechanism (dispatching). This lets a developer concentrate on coding policy. It also would allow different applications to communicate with their preferred policy to bargain for scheduling time. Custom schedulers are special Windows NT drivers that coexist with the default NT scheduler, and which should have negligible impact on global performance.

Candea then reviewed the current state of Windows NT scheduling. The basic schedulable unit is the thread, which acquires CPU time based on priority levels of two classes: variable, which uses a dynamic priority round-robin, and real-time, which uses a fixed priority round-robin. Interrupts and deferred procedure calls (DPCs) have precedence over threads, so scheduling predictability is

limited. Scheduling events are triggered by: the end of a thread quantum, priority or affinity changes, transition to wait state, or a thread waking up.

NT timers use the hardware abstraction layer (HAL), which provides the kernel with a periodic timer of variable resolution. Candea noted that most HALs have resolution between 1 and 15 ms, but some HALs are worse than others – some can be set to only powers of two, while others are fixed at 10 ms. This is certainly another limitation to scheduling with any policy under NT.

Vassal separates policy from mechanism. While the NT scheduler consists of thread dispatch and a default scheduler, Vassal consists of many schedulers (policy modules) arranged hierarchically and a separate dispatch module in charge of preempting and awakening threads. Standard NT policies remain in the kernel so that applications with no special needs are handled as usual.

The schedulers register decision making routines with the dispatcher. The dispatcher invokes these when scheduling events occur. Threads can communicate with schedulers to request services. These new features require some interface modifications, with the addition of three system calls.

As proof-of-concept, the Vassal team implemented a sample scheduler that can be loaded in addition to the default NT policy. The sample allows threads to get scheduled at application-specified time instances, which is simplistic, Candea admitted, but demonstrates potential for more interesting time-based policies.

Minimal NT kernel changes were required: they added 188 lines of C code, added 61 assembly instructions, and replaced 6 assembly instructions. The scheduler itself was only 116 lines of C code, required no assembly language, and they only needed to code policy. Performance results showed that it was no slower if no specialized scheduler was

loaded, and there was only 8% overhead with their untuned prototype. Results showed that with the special scheduler, the predictability of periodic wakeup times significantly improved, and there were no longer early wakeups. There were a few slightly late wakeups (still less late than without the prototype loaded), and these were caused by unscheduled events such as interrupts and DPCs.

Candea emphasized the Vassal take-homes are that it demonstrates the viability and positive impact of loadable schedulers, and that it frees the OS from anticipating all possible application scheduling requirements. It also encourages interesting research in this area by making it easy to develop and test new policies, and doesn't adversely affect the OS.

Some related work is Solaris, which maps scheduling decisions onto a global priority space; extensible OS work like Spin, Exokernal, or VINO, and hierarchical schedulers like UTAH CPU or UIUC Windows NT Soft Real-Time scheduler.

Many questions followed. Someone suggested that two different schedulers must be compatible or there will be trouble. Candea agreed that this was an interesting problem that could be solved by allowing schedulers to talk to each other and "negotiate" or to use their descriptions and decide if a conflict will occur. Another question was that a driver has limited visibility into the NT kernel, and does this affect the power of a scheduler? The answer is yes, ideally these special drivers would be able to see into the kernel data structures for real power. The moderator asked what can be done about predicting DPCs and interrupts. Candea didn't think that was necessary, noting that these are best left to perform their crucially important tasks when they need to.

<<http://pdos.lcs.mit.edu/~candea/research.html>>
<<http://research.microsoft.com/~mbj>>

Session: NT Futures

Tom Phillips and Felipe Cabrera,
Microsoft Corporation

Summary by Kevin Chipalowsky

Felipe Cabrera and Tom Phillips demonstrated the upcoming Windows NT 5.0 operating system and the NT Services for UNIX add-on. The two-hour session was very open, informal, and at times emotional for some in the audience. Microsoft gave a presentation while inviting just about any type of question regarding the future of their flagship operating system. Cabrera and Phillips fielded very spirited questions and comments.

Phillips began his NT 5.0 presentation by describing its new support for upgrading from Win 9x. The setup program first scans a system for compatibility before installing anything. It will now migrate applications as part of the upgrade process, using plug-ins to support third-party software. The system configuration is also preserved during the transition to NT.

Next, Cabrera talked about the new volume-management infrastructure. The new version of Windows NT will have many new storage management features. In most cases, hard-drive partitions can be manipulated without requiring a reboot. For example, partition size can grow and shrink dynamically. There are also new security features, such as a reliable "change" journal and file encryption. In response to a question about the type of encryption, Cabrera explained that it uses public key cryptography and is designed to prevent thieves from examining data on a stolen laptop.

Cabrera also talked about the new file-based services sported by NT 5.0. It provides a new content indexing tool, which can be used to search for files based on their content instead of just their filename. It tracks common types of embedded file links and updates them when a data file is moved to a different volume

or even a different machine. There is also a new automated recovery system to revive a computer that otherwise will not boot.

The speakers then presented NT Services for UNIX. Microsoft developed it in response to the growing adoption of NT by previous UNIX users. It will be available for Windows NT 4.0 with Service Pack 3 for \$149 per client. It is in beta, and anyone interested in being a beta tester should email <gregsu@microsoft.com> with "sfubeta" as the subject.

NT Services for UNIX will make an NT machine feel more like a UNIX one. It allows users to access NFS partitions like any NTFS or FAT partition. A new Telnet client and server are also included, so administrators can remotely Telnet into a Windows NT system and run console-based applications. Microsoft has licensed a Korn shell implementation and a few dozen familiar UNIX tools from MKS. The audience loudly applauded this part of the presentation.

Next, Cabrera and Phillips revealed storage features; the new Microsoft Management Console (MMC) was the center of attention. It is a management container that provides Microsoft and third-party developers the opportunity to plug in software to manage just about anything.

The first demonstration was of RAID 5.0 support. One hard drive of a stripe volume was removed and later plugged back into the system. Although the underlying file system seemed to handle the intentional fault, MMC simply crashed and the computer needed a reboot. After this small setback, the demo moved on.

Hierarchical Storage Management (HSM) is another feature new to NT 5.0. It makes use of the observation that the most commonly used files are usually the most recently used files. When a hard-drive partition becomes full, the filesystem offloads older files to a tape backup system to make free hard-drive space

available. Although Microsoft is not the first to attempt to build HSM support onto NT, they believe they will be the most successful. They have complete control over the operating system and can fix all of the related utilities that would otherwise have difficulty with the extremely long latency that results from trying to open certain files.

Networking in NT 5.0 has also received an overhaul. As with volume management, most network configuration changes can now be made without rebooting the system. Microsoft also claims an improved programmable network infrastructure. The TCP/IP stack is also enhanced; it runs faster and supports security and QoS protocols.

Large Installation System Administration of Windows NT Conference

SEATTLE, WASHINGTON,

August 6-8, 1998

INVITED TALK

Windows Management Roadmap

Nikhil Joshi and Tom Phillips,
Microsoft Corporation

Summary by John Holmwood

This talk, which was intended to provide a roadmap of the changes Microsoft is making to the manageability of Windows NT, was broken into two parts. Tom Phillips talked about the Windows Management Architecture (WMA) and Nikhil Joshi discussed the NT File System (NTFS).

The WMA provides the management framework and structure to assist system administrators in managing Windows NT. It includes the suite of applications formerly called Web Based Environment Management (WBEM). It is based on the

Windows Management Instrumentation (WMI) object model. Phillips demonstrated several applications that use the WMI:

Disk performance. Phillips demonstrated the WMI object model using a script program developed by Computer Associates. He noted that supporting the WMI interfaces is a requirement for getting device drivers certified for NT 5.

Microsoft management console. Phillips demonstrated the Common User Interface in the new management console. He noted that it was possible to group tools appropriate for a particular management function, such as database management, and save the tool set so that the administrators will have a specific set of tools appropriate to their function.

Windows Scripting Host. Phillips demonstrated the new scripting architecture, Windows Scripting Host, by resetting all network adapters in a sub-net without using the GUI. This demo drew cries of "about time" from the audience.

Automatic software installation. Using the new Policy Manager, Phillips created a policy to have an application automatically upgraded on a user's desktop. He used the Active Directory to find the file share that contained the new application. When he logged on as that user, the new application was automatically downloaded to the workstation. This drew questions regarding license management and network bandwidth requirements, which he dodged.

At this point, Nik Joshi took over the presentation. He provided some historical background on the evolution of Microsoft filesystems, then talked about the new NTFS. The biggest news was the changes between NT 5 and NT 4 NTFS. The NT 5 installation process converts the old NTFS to the new one automatically. Once the file system has been converted, it cannot be converted back to the NT 4 version. This means that the NT 5 NTFS should not be used on a dual

boot (NT 4/NT5) computer. NT 5 should be installed on a separate machine.

NT 5 has incorporated the Veritas Logical Disk Manager and Eastman HSM, as well as file encryption and disk quotas. Joshi provided three demonstrations:

Volume manager. Joshi first demonstrated the Disk Manager application. The capabilities will be familiar to any UNIX system administrator; all the Unices I know now include the Veritas Logical Disk Manager. However, judging by the audience reaction, NT sysadmins will appreciate the functionality.

Plug and play. Joshi demonstrated the new plug and play capabilities by installing a PCMCIA NIC into his laptop while it was running. The system detected the new card and loaded the correct drivers and protocol stack automatically. It simply worked before our eyes.

Resource Kit. In order to demonstrate the tool set incorporated into the NT 5 Resource Kit, Joshi demonstrated the Nettest tool, which he characterized as Ipconfig on steroids. Microsoft has done significant work to make the Resource Kit easier to use.

REFEREED PAPERS

Session: Management and Monitoring

Summary by Chris Barnash

Patch32: A System for Automated Client OS Updates

Gerald Carter, Auburn University

Patch32 was created to "provide for completely automated, remotely administered updates to Microsoft's 32 bit operating systems." In addition, the goals for Patch32 included the ability to support Windows 95 and Windows NT with the same update method, and the ability to provide an update method that is free.

The Patch32 system is made up of two main components, the server and the client. The server in Carter's implementa-

tion consists of Samba running on a Sparc Ultra 170. On this server is a share that provides a central location for the distribution of client updates. The client side of Patch32 consists of a Perl for Win32 script, which is executed by the client during the boot process.

The Patch32 Perl script is charged with several tasks. First, it determines what version of the operating system is running on the client. On the basis of this information, it determines the location of the patches on the server, and the method of installation. (The installation method differs between Windows 95 and Windows NT.) Next, it parses a file containing a list of patches stored on the server to determine which patches are available for installation. For each patch in the list, the Patch32 Perl script queries the client's registry to determine if the patch is already installed. If it is not installed, it installs the patch, otherwise it continues down the list. Upon completion, a message is displayed providing information on the installation. If the client is Windows 95, the patches will take effect after the next reboot. If the client is Windows NT, the Resource Kit's shutdown utility is used to reboot the system.

Documentation and source code for Patch32 can be found at
<<http://www.eng.fsu.edu/users/cartegw/Patch32>>.

Monitoring Utilization in an NT Workstation Lab

Paul Kranenburg, Erasmus University, Rotterdam

Paul Kranenburg discussed his solution for monitoring usage in the computer labs at Erasmus University – a Windows NT utmp service.

The utmp service relies on NT's built-in auditing features to document LOGON and LOGOFF events. Specifically, SUCCESSFUL LOGON and SUCCESSFUL LOGOFF events are used to identify when a particular computer is being used.

The data gathered by the utmp service can be used in one of two ways. Short-term data provides for immediate notification of computers that may be down. An on-screen map displays all of the workstations with color codes that identify the status of the computers. A kiosk showing this map is also set up at the entrance of the labs to assist in finding available computers. The long-term data statistics can be graphed to provide utilization reports, which in turn are used to determine if the current number of workstations is sustaining the needs of the department.

The author can be reached at
<kranenburg@few.eur.nl>.

INVITED TALKS

Summaries by John Holmwood

Windows NT – A New O/S that Architecturally Isn't so New

Mark Russinovich, Systems Internals

Mark Russinovich maintains the Systems Internals Web site, a resource for Windows NT and Windows 9x utilities. He is also the author of the NT Internals column in Windows NT magazine. Russinovich opened his presentation with a short history of the development of UNIX and Windows NT, and then compared the core services of Windows NT and UNIX. (Russinovich carefully pointed out that he was talking about the NT exec, not the Win32 APIs that are layered on top of the exec, and that he was talking about UNIX in general, not any specific implementation.) The areas of comparison were:

Architecture overview. The main architectural difference is that UNIX does not have a HAL. Windows NT is so much like VMS that it is possible to follow what NT is doing internally by using the VMS internals documentation. He provided a Rosetta Stone for translating VMS documentation to NT.

Namespace. The Object Manager defines NT's namespace. This makes the namespace uniform. The UNIX namespace is defined in terms of the filesystem. It doesn't need an Object Manager. Russinovich believes the NT method is superior.

Process management. NT process management includes processes, threads, and a scheduler. The NT kernel mode is fully preemptive. In this category, the Unices vary significantly. A UNIX process is similar to an NT file handle. The kernel is cooperatively preemptive.

Memory Management. NT and UNIX are similar here.

Security. Both NT and UNIX are rated C2 secure systems. NT uses ACLs, users, and groups. The groups are nestable. There are about 20 different privileges. Security is handled by the Object Manager. UNIX has a simpler security model based on Users and Groups. ACLs have been added to some versions. Security is applied to files. This difference is due to the differences noted in the namespace section.

Synchronization and IPC. Similar. Russinovich moved over this area quickly.

I/O. NT I/O is centered around the file object. This allows a layered driver architecture that can support asynchronous operations including hardware interrupt support. Plug and play capability is coming in NT 5. UNIX I/O is centered around vnode/inodes. Traditional I/O is synchronous. Some versions have split interrupts to support asynchronous events.

File disk cache. NT has a single global cache. The virtual file cache is mapped into the kernel memory cache. UNIX uses disk block cache. Some of the newer versions use the same cache model as NT.

Networking. Lots of interfaces, lots of protocols. The difference is the layered model in NT. Only streams are layered in UNIX.

Integrated database. NT has a Configuration Manager Registry. UNIX uses config files.

Extensibility. In NT, all drivers are dynamic. There is a rich set of operating system APIs for drivers. The layered I/O allows drivers to add functionality. UNIX supports dynamically loaded extensions. The degree of operating system support services varies from very limited to a set approaching those provided by NT.

Portability. In terms of CPUs supported, UNIX is available on everything. NT is only available on x86 and Alphas.

Russinovich finished his presentation by tackling the question, "Which is better – UNIX or NT?" He put up charts of published Specweb and TCP-C benchmarks. His conclusion was that NT is as good as UNIX for small- to medium-sized servers and will get better in the larger-server space over time.

NT 5.0 Migration Strategies at Microsoft

Curtis Cummings, Microsoft Corporation

Curtis Cummings is responsible for IT support at Microsoft. He started work on Windows NT when it was the Cairo project. He is responsible for the rollout of NT 5 at Microsoft, which runs its entire company on Windows NT. He has 150 NT servers running the Beta 1 NT 5 software. The talk included a great deal of light banter between the speaker and the audience. Todd Needham of Microsoft fielded marketing questions for Cummings. Since Curt had a microphone and Todd didn't, this occasionally gave the impression of Curt acting as Todd's puppet.

Cummings started his talk by describing the Internal Technology Group's (ITG) environment, noting that his clients run "dog food," Microsoft's term for Alpha code. Two years ago, Microsoft didn't use DNS internally. In response to a question from the audience, Cummings noted that

only his new administrators use the Microsoft GUI interface to DNS. Everyone else uses the command line interface.

He then described issues in migrating to NT 5 using the ITG experience as an example.

Use what you've got. Most of your infrastructure can continue to be used. You will probably have to beef up the server hardware. Cummings shared the ITG Network Plan to test NT 5 RAS services. When asked if Microsoft would be sharing the results of the ITG test, Todd stated that they would be publishing their acceptance test criteria.

Pick a migration approach. Cummings described ITG's migration plan in detail. The schedule has slipped. His slides showed ITG's rollout completing by the end of 1998. This has been changed to coincide with the official release of Windows NT 5. They plan is to have all of their servers using NT 5 in production when NT 5 is officially released.

Namespace design. Cummings devoted a quite a bit of time to the new DNS-like domain structure that Microsoft has set up for NT 5. He believes this is one of the biggest changes for people who are used to WINS.

Tools. According to Cummings, the Microsoft Management Console (MMC) is "a way cool tool." On the other hand, he uses SMS because he is not given any other choice. ITG had to build their own tools for managing NT 5. The MMC gives them a consistent interface for the tools they developed. Microsoft sent the ITG staff to Perl courses, and now most of their internal tools are written in Perl. There was a lot of support for an audience request to have access to the ITG tools.

Policy manager. Cummings believes that the policy manager in NT 5 is a "big deal" for the support staff in terms of both administrative support for it and

planning and definition of appropriate policies.

Planning your Infrastructure. Cummings talked about how to plan your server infrastructure. Bandwidth requirements will be the major issue. There was a suggestion from the audience that ITG should do its bandwidth testing on the India link (ITG's slowest link, 64KB).

Migration Order. Microsoft's migration order looks backward; they're doing the most critical components first. This is required to initiate the new services. Their fallback position is "God help us!"

Security. A member of the audience from MIT confirmed that Microsoft is working closely with MIT to make Microsoft's Kerberos interoperate with the standard implementation.

Bringing the "Real" Internet to Windows NT

Bo Ahlberg, MetaInfo, Inc.

Bo Ahlberg was the chief technology officer at MetaInfo. MetaInfo has ported the IETF reference version of BIND and sendmail to Windows NT. MetaInfo was recently acquired, and Ahlberg is not staying with the new organization. He noted that the engineer who actually did the port wasn't allowed to do the presentation, so we were stuck with him. His talk was subtitled "Making NT into a Real 'Forking' OS."

The first third of this talk was on the general problems MetaInfo encountered in porting UNIX applications to NT: a fork is not a thread, a file descriptor is not a handle, there are no common tools, and a daemon is not a service.

He then briefly described porting tactics:

Start from the beginning and design with the other architecture in mind.

Fix what you've got; this teaches you your design weaknesses.

Starting over is sometimes cheaper than using what you've got; you can design in portability.

In the final section of the talk Ahlberg described the port of BIND and sendmail – or, more accurately, the mistakes, the successes, and the lessons learned.

BIND. They created a service wrapper for BIND to make it fit the NT architecture, and modified the error code to use the Event Log. This allowed the BIND daemon to run as a child process. The lessons learned were that NT-ifying UNIX code is wrong; maintaining compatibility with the "owner" of the code is a "good thing"; you shouldn't mix MFC with services; and sometimes it's easier to fix the environment than it is to fix the program.

sendmail. Sendmail would not work without a fork architecture, which NT didn't have. After several tries, the project team created a fork environment for NT.

Ahlberg ran out of time before competing all of his slides. He rushed through the conclusion that it is possible to port UNIX applications to NT but the work needs to be planned and scoped very carefully; then you need to "UNIX-ify" NT, not the reverse.

PANEL

Windows NT Tips and Tricks

Robert O'Brien, Microsoft Corporation; Brian O'Neil, Mike Wei, and Andie O'Brien, Collective Technologies

Summary by Chris Barnash

The Windows NT graphical user interface often leads system administrators to believe that it is impossible to run headless, remotely managed, NT servers. But according to Robert O'Brien, it is possible to deploy NT servers in this fashion.

O'Brien's talk, entitled "Windows NT Lights Out Operation," focused on the setup and update of remotely managed NT servers. O'Brien outlined five major steps for deployment: (1) Choose a system console solution. Several hardware vendors, such as Compaq, DEC, HP, and

Intel, offer Integrated Remote Console Boards. (2) Choose a telnet/secure shell solution. Microsoft is working on Services for UNIX, which will include this. Also, several third-party vendors offer telnet/secure shell services, including Seattle Lab and DataFellows. (3) Choose a remote Win32 solution, such as Carbon Copy (Compaq), PCAnywhere (Symantec), Remote Possible (CA Associates), and Virtual Network Computing (ORL). (4) Develop an OEM-unattended installation process. Two options exist for this step. Microsoft has developed a process for unattended installs, which can be found at <http://www.microsoft.com/networkstation/ntwnew/info/deployguide.htm>. The other option is to use a disk cloning technique, like Ghost. (5) Choose a network management/monitoring solution. This can be accomplished with Microsoft Systems Management Server, Tivoli, Computer Associates Unicenter TNG, or Hewlett Packard OpenView.

O'Brien's paper on the Windows NT Lights Out Operation, slides from the presentation, and tools to assist administrators with deployment can be found at <http://ftp.msftlabs.com/loop>. Additional information on Windows NT management can be found at <http://www.microsoft.com/management>.

Brian O'Neil discussed Windows NT Terminal Server, a redesign of Windows NT Server backend that supports thin clients. Windows NT Terminal Server can provide the use of Windows NT on clients that cannot run Windows NT. The idea is very similar to Xterms in the UNIX world. Thin clients (such as a 386) can run the Windows GUI without the need to actually run the full-blown Windows NT on the desktop. More information on Windows NT Terminal Server can be found at <http://www.microsoft.com/NTServer/Basics/TerminalServer/default.asp>.

Mike Wei discussed UNIX and Windows NT filesharing with respect to interoperability, performance, security, and namespace consistency. The most interesting

part of the discussion was that of Microsoft's Distributed File System (DFS) for Windows NT. DFS is similar to NFS under UNIX. It allows an administrator to set up mount points instead of shares assigned to different drive letters. DFS will be included in Windows NT 5.0 and is available as a download for Windows NT 4.0. For more information on DFS, see <http://backoffice.microsoft.com/download/moreinfo/dfs.asp>.

Andie O'Brien discussed several mechanisms for monitoring multiple NT servers. The Performance Monitor can be used to monitor several aspects of a computer, including processor, memory, and page file utilization. Another built-in NT tool is the Event Log. Keeping an eye on the logs can help pinpoint problems. O'Brien also gave pointers to several third-party monitoring programs from HP, NetIQ, and Heroix.

INVITED TALK

NT 5 Administration: Change and Configuration Management

Dan Plastina and Mike Cherry,
Microsoft Corporation

Summary by John Holmwood

This presentation, the final session of the conference, was meant to be a demonstration and discussion of the behind-the-scenes technology that supported the new Change and Configurations Management in Windows NT 5. Predictably, most of the demonstrations did not work the first time.

Unfortunately, most attendees missed some or all of the talk, which was liberally interspersed with lively audience discussion after each demo. In fact, Remy Evard, co-chair of the conference, had to ask the audience to hold off on questions so that the formal part of the talk could be completed in two hours. When I left after three hours to catch my plane home, Plastina and Cherry were still fielding questions from the few people left in the audience.

NetPC boot. The first demonstration was the NetPC network boot. This failed the first time they tried it, but eventually, with Cherry working in the background while Plastina talked, they managed to get it working. The functionality requires a special NIC card that supports the PXWE protocol (NetPC & PC98) on the client and the new Active Directory, DNS, and DHCP services on the server. The functionality is automatic with NT 5 DHCP; other DHCPs should work with the new NIC cards.

The boot sequence starts by running fdisk on the client hard drive. Dual boot configurations are not compatible with this feature. Plastina solicited feedback from the audience on the need to support dual-boot systems; the response was mixed.

Application management capabilities.

During this demonstration, Cherry was able to automatically install an application by selecting the application. However, his attempt to demonstrate that the application could be installed simply by invoking a document created by the application failed.

A request came from the audience for this capability to be included in NT 4.5. Mike and Dan pointed out that half the problem was in the client and half in the server. On the server side, the solution relies heavily on NT 5 technologies such as Active Directory, Kerberos, and caching. There is little chance of these functions appearing in an NT 4 service pack.

Policy management. The policy management user interface is likely to change before NT 5 ships. Policy will become part of the property of the container. Much of Policy Editor functionality is not scriptable. (Plastina regrets this decision). Policies are more consistent than in NT 4, but user permissions aren't interfered with.

Synchronization manager. The synchronization manager manages the client-side cache. Plastina tried, unsuccessfully, to demonstrate the system synchronizing a file. He did demonstrate the synchronization options available on the system. The synchronization function requires Windows NT 5 server as the fileserver. It uses the SMB redirector so won't work with NFS or Novell. The speakers dodged the question of testing the functionality with Samba.

Roaming profiles. Plastina commented on the effort that has gone into Office 2000 to make it an "awesome" roaming application. The application now understands the difference between user data (e.g., my dictionary) and application data. During the questions after this demo, a lot of hostility regarding roaming profiles came up. This appeared to be a case of killing the messenger. Plastina handled the shots very well. Roaming profiles are useful for the segment of user environments where users can add their own applications. For more locked-down environments, policies in NT 5 can redirect where files are obtained from without using roaming profiles.

There was also a lot of hostility over Microsoft not following its own application guidelines with respect to DLLs. Plastina's response was that Microsoft application groups could no longer change O/S DLLs. This will eliminate some of the problem of applications interfering with each other.

Plastina would really like samples of real login scripts so that Microsoft can understand what workarounds for NT 4 people are using. This will help make the NT 5 functionality better. Send scripts with commentary to <danpl@microsoft.com>.

Third USENIX Workshop on Electronic Commerce

BOSTON, MASSACHUSETTS

August 31-September 3, 1998

Session: Advances in Payment Technology

Summary by Matthew Hohlfeld

Electronic Commerce and the Street Performer Protocol

John Kelsey and Bruce Schneier, Counterpane Systems

Bruce Schneier began his presentation by letting the audience know that, through his error, the version of the paper in the proceedings was not the correct one, which is available at <<http://www.usenix.org/publications/library/proceedings/ec98>> and at <http://www.counterpane.com/street_performer.html>.

The premise of this work is that none of the currently available systems for protecting intellectual property (IP) will be a full solution in a world where the duplication of a work is simple and nearly cost-free. As technologies for copying become cheaper, and we begin to rely more on digital delivery systems for intellectual property, the notion of copyright itself becomes invalid. Without copyright protection, the creation of works will diminish, and ultimately the public will suffer. As a possible solution to this problem, Schneier suggested that new works could be funded on an escrow system. This would help restore the incentive for the creation of works, even when the works are then released as part of the "public good." He provided an exhaustive list of the currently used (or designed) methods for protection of IP, along with the perceived flaws of each. The list included both secure perimeter schemes and traitor tracing schemes; law-based solutions; and alternatives such as advertising, product placement, and government funding.

By suggesting an analogy between "consumers" of IP and donations to a street performer, Schneier provides the basis for a new payment method, dubbed the Street Performer Protocol. Using a bank or publisher as a trusted third party, the Street Performer Protocol simply has the creator of a work request a specific minimum amount of donations. Once that level is reached, the creator promises to release the work into the public domain. After releasing the work, the third party transfers the donated money to the creator.

After discussing some of the motivations that individuals would have to become donors, and the fact that the whole notion of IP is relatively new, Schneier opened the floor to questions. Perry Metzger suggested that this protocol/economic model would not result in the same volume of goods as the market can sustain, and pointed out that Penguin Classics makes money as a value-added reseller of works in the public domain. Max Tsvetovotat inquired as to the applicability of this protocol to open source software; Schneier responded that the closest match would be to use proposed feature lists as the description of the "work." Stuart Feldman then pointed out that Victorian subscription incorporated a very similar economic model, in which the contributors' names appeared on the first page of each work. Nicko Van Someren pointed out that widespread use of this protocol could result in IP production focusing on form rather than substance, and that production could change course back to forms, such as books, whose reproduction is more difficult. Juan Garay asked whether it is a problem that the trusted third party will become quite large when dealing with works that are reviewed prior to publication; Schneier responded that this already occurs.

VarietyCash: A Multi-purpose Electronic Payment System

M. Bellare, University of California, San Diego; J. Garay, Bell Laboratories; C. Jutla, IBM T.J. Watson Research Center; M. Yung, CertCo

Charanjit Jutla presented the culmination of some six years of cooperative research by the authors into a form of digital cash intended to replace physical cash. They explicitly consider only the single-issuer situation, but assert that the scheme can easily be extended to multiple-issuer. The main problem addressed by their approach is the need to incorporate network-based (online) and smartcard based (offline) solutions in a single interoperable system.

After discussing some other electronic payment schemes, and how they fare with regards to anonymity, atomicity, and network vs. smartcard issues, Jutla introduced the features of VarietyCash. These include an Issuer that maintains a list of all issued but unspent coins and a truly account-less system for ease of maintenance. The main cost of the system is incorporated in verification of each transaction. The Issuer in this system is trusted to preserve anonymity (which can be removed if required, e.g., by law).

Some of the security goals addressed by VarietyCash are: protocol security, internal security, user security, and network security. It uses tamper-proof hardware at the Issuer site to prevent insider attacks. Storing an encrypted version of each issued coin in a database at the issuer site is also required as a defense against insider attacks. The main goal in User Security is to prevent the loss of the user's anonymity.

Other features of the system include: using the issuer as a certification authority, a global requirement for "conservation of cash." Jutla also gave a brief trace through the Coin Purchase Protocol, emphasizing the speed of the Issuer in

this protocol, and then showed the Payment Protocol at a high level. He said that a prototype system has been built, and that anyone interested in a demo should contact him.

Raphael Yahalom asked about the difference between anonymity in this system and the trust in credit card purchases; the answer was the receipt is the basis for the accountability in credit card purchases, and there's no such receipt in this system. Nevin Heintze suggested that some requirements were missing from the formalization, including guarantees of coin value and coin delivery upon purchase. Mark Seiden questioned the value of using VarietyCash versus a ledger-based system; the benefit is in the efficiency of the VarietyCash approach. Seiden also wanted to know what the result of an issuer private key breach is; the answer was that there is a large consequence to this, and the system is built to insure that the private key is kept private. An offline conversation with Jutla provided this more complete response: the Issuer has many keys - one for signing the coin, one for encrypting the coin in its database, one that is its public-private key pair for communications, one that is a public-private key pair for certificates. The system doesn't break if any of these keys is lost. It may lead to small-scale attacks: for example, if the signing key is lost, it leads to insider attacks; if the database encryption key is lost, again it may lead to insider or intruder attacks; if the communication key is lost, it could lead to theft of users' coins.

NetCents: A Lightweight Protocol for Secure Micropayments

Tomi Poutanen, University of Toronto; Heather Hinton, Ryerson University; Michael Stumm, University of Toronto

Tomi Poutanen discussed NetCents, which was originally designed to be a micropayment protocol for use on the Internet. The protocol has since been extended to support a wider range of val-

ues while maintaining efficiency and robustness.

Part of the motivation for this work is the fact that credit cards are used as the primary payment method on the Internet, even though their use includes several flaws. Poutanen listed the desired features in a payment protocol for the Internet. It must: be secure (by avoiding double-spending), be goods atomic, support anonymity, and support a full range of values. Importantly, it must be implementable; thus, it must be scalable, interoperable, and low-cost. He also noted that to gain consumer support, it should be "as easy to use as real cash."

The design of NetCents considers the tradeoff between security and cost that is inherent in the online versus offline decision. Its heritage is traced to Millicent, but it is more feature-rich, secure, and cheaper than Millicent. NetCents makes use of the public/private split in several ways, including separation of the scrip, and using public-key verification to verify each electronic payment order.

Batching of payments in NetCents is used to increase efficiency and remove some load from the bank. This feature directly influences the need to be able to transfer scrip from one vendor to another; this is included as a sub-protocol of the purchase protocol. After describing these features, Poutanen discussed fraud control in their system. The protocol ensures that scrip transfers are atomic, and that though a criminal vendor can allow customers to double-spend, a probabilistic verification scheme makes it economically undesirable to do.

Poutanen then described how NetCents handles the crossover between online and offline protocols. Online arbitration is performed by means of a trusted third party, and the issuer maintains a certain minimal balance so that large purchases can go through while smaller balances are stored at vendors in support of cheap (offline) micropayments. The performance of an individual vendor depends

highly upon the scrip co-location hit performance.

Bill Frantz wanted to know if the need for secure hardware at the issuer's site indicated that we also need trusted code at each vendor's site; the answer was that NetCents doesn't require tamper proof hardware at each vendor, but only may catch cheaters. Manoj Kumar asked how to detect which of possibly many vendors in a scrip transfer chain was the malicious one; the answer was a full audit path is required to determine which was the malicious vendor. Kumar then pointed out that this would require each vendor to keep track of all outbound scrip transfers each day (to be flushed when transferred to real monetary value). Tracy Mullen asked what happens when a vendor fails during a purchase; the answer was that the value is rolled back into the scrip, as if the purchase had not occurred.

Session: Auction Markets

Summary by Matthew Hohlfeld

The Auction Manager: Market Middleware for Large-Scale Electronic Commerce

Tracy Mullen and Michael P. Wellman, University of Michigan

To introduce the motivation for their work, Tracy Mullen described an ongoing project in which the University of Michigan Digital Library is modelled as an information economy. Within this environment, different types of organizations have different market policies (e.g., corporate information policies vs. grade school ones). Participating individuals change over time and have needs that vary with current projects and interests. The authors' work addresses how to build flexible infrastructures that can support such dynamic, diverse economies. Mullen described how formal description languages for goods, services, and auctions (where auctions are really just services for setting a price) are used within their system interaction protocols and middle

ware components to automate and simplify the electronic commerce process.

The talk focused on a particular component, the Auction Manager, which generates and tracks markets and provides other market management services. Commerce on the Internet often faces different economics than those of more traditional commerce channels, including lower transaction and distributions costs and marginal costs near zero for information goods, which tends to promote product bundling and unbundling. The Auction Manager must be able to match buyers and sellers in this more complex environment.

By extending product descriptions to include bundling and unbundling operators, Mullen showed how logical inference rules could be used to locate appropriate markets for buyers and sellers, as well as find potential arbitrage opportunities across related markets. In addition, the concept of buyer's and seller's choice bundling was introduced. Buyer's choice means that instead of buyers buying an entire bundle of goods or services, they can buy the option to choose one or more of these products; seller's choice is similar. For example, a newspaper is really a bundle of articles sold as seller's choice; this same bundle of articles could be sold as buyer's choice, where the buyers decide which they want to read.

In addition to matching agents with markets, the Auction Manager also serves as the focal point for market creation and selection policies. Since infrastructure costs and agent decision complexity costs exist for each market created, unlimited market creation has the potential to overwhelm the system. The Auction Manager is being used to explore enforcing various market-creation policies, such as having explicit organizational policies or using auction fees to provide agents with the right incentives not to create unnecessary markets.

Finally, the Auction Manager can also serve as a repository of market and orga-

nization-specific knowledge about selecting the best kinds of markets for exchanging different types of goods or services. While agents can always specify exactly the type of market they desire, they also have the option of allowing the Auction Manager to fill in reasonable defaults. Future work includes extending both the market creation and market selection policies.

The only question came from Max Tsvetovat, who wanted to know if the Auction Manager is a central, trusted component of the system; the answer was that though it can be distributed, it is centralized and trusted for simplicity.

Internet Auctions

Manoj Kumar, Stuart I. Feldman, IBM T. J. Watson Research Center

Manoj Kumar started his presentation with an anecdotal note about the history of auctions on the Internet: the most expensive T-shirt sold from the Nagano Olympics Web site in a series of daily sealed-bid auctions went for \$15,000. He went on to present a list of the auction issues he would not be addressing: legal issues, cheating and/or sabotage, social issues, and multi-piece or continuous auctions.

With all of that out of the way, Kumar launched into predictions about the future of auctions on the Internet. The authors believe that there will be an increase in the popularity of auctions for several reasons, including the integration of auctions with existing business processes and an increase in trust in the auction model. Kumar noted that though there are already many traditional uses of auctions in business-to-business transactions, there is room for new forms and uses of auctions in business-to-consumer transactions.

Kumar then provided a classification of auctions and introduced some of the common variations on these types. He then introduced the system that the authors built on top of Net.Commerce

(an existing IBM product) to allow for the integrated use of auctions in business-to-consumer transactions. They built this in a consumer-oriented fashion and made it as close to fixed-price shopping as possible. By choosing to make this a consumer-oriented project, they became aware that the user interface is very important to its success. Kumar then described the user interface and its similarities to the “shopping cart” metaphor in current Web sites.

The Auctioneer is the commercial site in this setup, and the user interface for the Auctioneer has also undergone extensive work, including “experts” that provide pre-defined auction types. This is all built using a generic infrastructure that supports a wide variety of auctions, a prototype of which is available at <http://www.software.ibm.com/commerce/net.commerce>.

Doug Tygar expressed his concern about the ability of this model to address the latency of bids, especially in double auctions; Kumar acknowledged that this problem is currently being ignored by the system. Tygar was also curious about how “accidental” bids are prevented; the answer was that a registration mechanism helps to prevent this (a full description is in the paper). Giray Pultar then asked about the use of agents for bidding in this system; the answer was that agents aren’t yet allowed.

Electronic Auctions with Private Bids

Michael Harkavy and J. D. Tygar
Carnegie Mellon University; Hiroaki Kikuchi, Tokai University

Michael Harkavy, winner of the workshop’s Best Student Paper award for this work, began his presentation by skipping his prepared description of traditional auctions, since the previous two speakers had already thoroughly covered that topic. He then presented a double-barreled argument for the use of sealed-bid Vickrey auctions in an electronic setting:

sealed-bid auctions are beneficial since they hide the preferences of the bidders, and Vickrey auctions have nice economic and efficiency properties.

Harkavy then discussed some of the problems in situations where bidding preferences are leaked, including signalling and cooperation between bidders. He claimed that simply having anonymous bidding is insufficient to protect the privacy of a bid.

Harkavy described how to leverage verifiable secret sharing and secret computation in order to implement an electronic auction that preserves the privacy of bids. This includes addition and multiplication on shared secrets, where the computation and the intermediate results do not reveal the secrets. Using this technique, more complex functions (like the maximum function) can be computed on secrets.

One restriction that the desire for efficiency forces on this system is the need to encode all bids into a given range. The bidders encode their bids into a secret that is distributed among auctioneers (where only t of the n auctioneers can be malicious). The auctioneers then compute the maximum function digit by digit on their inputs, eliminating bidders without revealing their bid or their identity. The winning bidder is revealed at the end of the computation.

Harkavy concluded his presentation by describing some of the lower-level details and optimization techniques used to attain some measure of efficiency. There is an inherent tradeoff in this system between the efficiency of the computation and the security of the individual bids.

Session: Secure Systems – What It Takes

Summary by Kevin Fu

A Resilient Access Control Scheme for Secure Electronic Transactions

Jong-Hyeon Lee, University of Cambridge

Jong-Hyeon Lee presented a way to authenticate customers without disclosing customer secrets to a merchant. (Lee, a student of Ross Anderson, is also capable of security in another dimension – Aikido.)

Despite the vulnerability to copying, passwords and Personal Identification Numbers (PINs) commonly authenticate customers to service providers. Lee sought a simple and secure electronic transaction model that does not have to explicitly transfer customer secrets and/or use public key cryptography. A scheme by Needham to control PINs is simple, provides for privacy, separates capabilities, and is customer-oriented; However, it is susceptible to replay attacks and bogus ATM machines.

Inspired by Needham’s scheme, Lee developed a customer-oriented transaction model in which the customer generates and maintains personal secrets. The model enables a transaction procedure among three principals: a customer, a merchant, and a bank. Principals can participate in registration, transaction, or secret-revocation procedures. A somewhat lengthy protocol explains the communication among the principals. By using only hash functions, Lee’s model enhances privacy for the customer and ensures nonrepudiation.

The registration procedure mimics that of Needham’s scheme, and the transaction procedure uses a technique from KryptoKnight. In Lee’s online scheme, the customer is involved with all procedures. An offline scheme works in a similar manner, but there is some extra com-

munication between the merchant and customer.

Asked whether there exists an implementation, Lee explained there is yet no implementation for this scheme, but there is for Needham's scheme.

See <<http://www.cl.cam.ac.uk/~jhl21>> for more information.

Trusting Trusted Hardware: Towards a Formal Model for Programmable Secure Coprocessors

Sean W. Smith and Vernon Austel,
IBM T.J. Watson Research Center

Sean Smith presented his findings on proving the security of secure coprocessors with respect to Federal Information Processing Standard (FIPS) 140-1 level 4 certification. His group worked on three goals: achieving level 4 certification as a research project, verifying the soundness or finding the holes in the coprocessor, and formally describing the coprocessor.

FIPS 140-1 specifies security requirements for cryptographic modules. The most stringent level in the standard, FIPS 140-1 level 4, requires a formal model of a system and formal proof of security. As of this writing, level 4 is an unachieved grail.

A secure coprocessor is hardware that must survive in a hostile environment. It must guarantee that memory contents will be zeroized upon any foreseeable attack, and it needs to defend against threats such as changes in voltage, temperature, and radiation. Such a programmable device is useful for e-commerce.

A mechanical theorem prover was iterated over a logical abstraction of the coprocessor. First, a formal model was made from a finite state machine. Then a specification was written in LISP to prove simple properties of security. The proof must show that the coprocessor maintains its security guarantees despite hardware failures and hardware attacks. Guarantees for security fall into three categories: safe execution, safe access, and

safe zeroization. Other assertions include authenticated execution, recoverability, and fault tolerance. The proof involves 2000 lines of C, 75 execution states, and 7500 lines of a mechanical proof.

Right now, only the hardware and bootstrap are being submitted for level 4 certification. IBM's plans for actual certification are still undecided. In this research, IBM went through a lot of the legwork for the bootstrap layer as an exercise; Smith notes that it would be "really cool" to go all the way with it. In the future, Smith hopes to evaluate the programs on the coprocessor. However, he expects complications, since the hardware could interrupt the software and the software could start interrupting the software.

Pointing out that FIPS is aging, an audience member asked Smith to share hints on where FIPS is falling short and where it goes too far. Smith replied that on the too-stringent side, FIPS requires the use of DSA for signatures. Everyone wants to use RSA, but to be FIPS-compliant, the coprocessor must contain algorithms no one wants to use. On the other hand, FIPS does not address security requirements of the manufacturing process.

Another audience member brought up the topic of differential power analysis with current fluctuations. Many security attacks result from crossing levels of abstraction (power analysis, buffer overrun, etc). Smith was ambivalent on whether good proof techniques can capture these attacks.

For more information, see <<http://www.ibm.com/security/cryptocards/>> and the IBM 4758 product brochure G325-1118.

On Secure and Pseudonymous Client-Relationships with Multiple Servers

Daniel Bleichenbacher, Eran Gabber, Phil Gibbons, Yossi Matias, and Alain Mayer, Lucent Technologies, Bell Laboratories

Alain Mayer talked about Janus, a cryptographic engine to establish and maintain pseudonymous relationships. Mayer enjoys hacking JavaScript and having fun on the Web. Coincidentally, he used the same Microsoft clip art in his presentation as does the Crowds project.

Janus facilitates relative pseudonymity. That is, a client is anonymous with respect to the client population (e.g., an ISP customer base). The server knows a message came from a particular client population, but it does not know which member of the population. Janus also allows for persistent relationships between clients and servers. Weak or strong authentication by means of passwords or keys allows for repeat visits.

Absolute anonymity is hard to achieve without a penalty in ease of use and performance. The work on Janus is complementary to other anonymizing efforts and can be combined with other techniques.

There is a distinction between data anonymity and connection anonymity. In data anonymity, data flowing over a connection does not reveal an identity. In this case the adversary would attack server endpoints. In connection anonymity, the connection itself does not reveal an identity, and the vulnerability is traffic analysis.

There are several candidate Janus functions. Mayer has three requirements of the function. First, it must ensure uniqueness of aliases among clients and resist impersonation; in other words, it must be hard to find an input that results in the same alias. Second, the function must not reveal information about clients. Third, there must be forward secrecy and statelessness for client mobil-

ity. Mayer described one such function involving a password-keyed hash of a client identifier, server identifier, and a usage tag. Mayer finds the CBC-MAC approach more promising than a simple hash because secrecy under a chosen message attack implies secrecy of passwords. The CBC-MAC approach fulfills all three requirements.

Janus works with email aliases. Aliased email can also help filter junk mail. A client may have a different mailbox for each server. One can filter (even by a third party) by ignoring mail to a particular alias.

Mayer indicated several places to house a Janus engine. In a local approach, the Janus engine lives in the client. Aliases would be routed through a proxy. This minimizes outside trust and cooperates with mobile code and Personal Privacy Preferences (P3P) repositories. In a gateway approach, a client need not download software. This allows easy upgrades and maintenance. In a third party approach, the Janus engine would exist in the outside world. The third party preserves subnet anonymity. Mayer pointed out that if you look at a gateway or local approach, the domain name or IP address does not reveal its alias or real address. A vendor could ask for a credit card for identity validation.

An audience participant asked whether anonymity is really beyond research and useful in the real world. Mayer responded that according to surveys on electronic commerce, end users worry about privacy. A high percentage of users leave sites that present fill-out forms. To demonstrate practicality, Mayer offered the example of personalized Web pages. A user no longer must remember passwords for services such as My Yahoo or NYT. Janus can be a tool to make personalized sites as easy to visit as regular sites.

The Lucent Personalized Web Assistant uses a Janus engine. See <http://pwa.com:8000/> for more information.

Secure WWW Transactions Using Standard HTTP and Java Applets

F. Bergadano, Università di Torino, Italy; B. Crispo, University of Cambridge and Università di Torino; M. Eccettuato, Università di Torino.

Francesco Bergadano presented an alternative for securing HTTP transactions. This solution uses local Java applets on the client side to establish a secure link with the server.

Existing solutions include modifications to the application protocol (e.g., SHTTP), a secure transport below the browser (e.g., SSL/TLS, DCE-Web transport APIs), proxy-based services, and network layer changes (e.g., IPsec). Bergadano's group wanted to achieve privacy, authentication, and possibly non-repudiation. However, they did not want to implement a new browser or modify existing browsers. Moreover, they wanted to provide strong cryptography and make the source code freely available.

The proposed architecture uses normal HTTP, TCP, and a Java-capable browser. Essentially, the client runs an applet from the server. This applet triggers a local applet that communicates with a local application on the client. This application in turn creates an encrypted channel with the server.

This approach requires relatively few changes. More important, Bergadano claims it does not require trust of the browser. It is desirable to separate security routines from the browser. This approach is similar to a proxy-based approach. However, a proxy must intervene with all communication. Bergadano's approach only becomes active when an HTTP transaction is explicitly asked to be secure.

Launching several questions, Avi Rubin asked Bergadano to answer just one: Where did you put security? Is it better than SSL, why can't you run a simple proxy? Are you assuming you can change

a firewall configuration? Taking a deep breath, Bergadano jokingly asked when dinner was scheduled. He chose to answer the SSL and firewall questions. In the case of SSL, one needs a trusted browser that supports SSL. In Europe, one cannot easily obtain a standard browser with strong cryptography. As for the firewall, Bergadano reported that the implementation was run on an open network. He was unsure about interactions with a firewall since a secondary channel must be established between the client and server. Another attendee commented that if this approach gets well used and works, it would be consumed by a browser.

For more information and the source code, see <http://security.unito.it/>.

SWAPER00: A Simple Wallet Architecture for Payments, Exchanges, Refunds, and Other Operations

Neil Daswani, Dan Boneh, Hector Garcia-Molina, Steven Ketchpel, and Andreas Paepcke, Stanford University

Neil Daswani presented the SWAPER00 digital wallet project. Started in September 1997, this project aimed to identify desirable wallet properties and features, define a wallet interaction model, define clean APIs for a wallet and its components, and build a prototype.

Daswani's group decided that: (1) A wallet architecture should be extensible. Rather than being completely proprietary, it should support multiple instruments and protocols. (2) It should not rely on a Web interface as the sole common interface. The basic architecture should be written once to be run anywhere. This enables the use of alternative devices such as Personal Digital Assistants (PDAs). (3) Symmetry allows for common services across commerce applications. Current wallet implementations are often non-symmetric; little infrastructure is shared between the client and server sides. (4) A wallet architecture should be

client-driven. The user should initiate all transactions. Vendors should not be capable of automatically invoking a client's digital wallet. After all, would you want a vendor reaching for your wallet as soon as you enter a store?

Daswani described a wallet interaction model. After starting a transaction, wallets can negotiate on a protocol. Because of symmetry, the user and vendor have similar wallets.

SWAPEROO has been implemented in C++ (PalmOS) and Java (Windows). Future work includes populating the wallet, experimenting with other devices (e.g., smart cards), working on the architecture, and abstracting out the data manager.

One question was asked about symmetry. Since everyone would have wallets of a similar design, is there any reason clients would not want to communicate with each other? Daswani responded that there are no restrictions. Another question involved tamper resistance. Given that the wallet must be in some tamper resistant memory, how are these things initialized? Daswani answered that for PalmPilots, this is a problem. However, in the future with a JVM access control, a policy could potentially be downloaded directly into the wallet from a trusted site.

A related paper on the PalmPilot implementation will appear in the future. The PalmPilot implementation lets a user buy a food item from a particular vending machine at Stanford. For more information, see <http://www-db.stanford.edu/~daswani/wallets/>

The Eternal Resource Locator: An Alternative Means of Establishing Trust on the World Wide Web

Ross Anderson, Vaclav Matyas, and Fabien A.P. Petitcolas, University of Cambridge

Vaclav Matyas presented an alternative means of managing trust in electronic

publishing. He spoke about WAX, a proprietary hypertext system for medical publishing. WAX uses hashes in combination with HTML links as an Eternal Resource Locator (ERL). Matyas is also the co-editor of the Global Trust Register, a massive directory with its own rating scheme of "top-level" PGP keys and X.509 certificates.

In the hierarchical WAX system, there are shelves owned by publishers, books owned by editors, and chapters owned by authors. WAX must protect against several threats: book contents could be altered, an incorrect book source could be claimed, or a publisher or author could deny past content. Matyas stressed that there are no confidentiality or audit requirements, only integrity and authenticity.

The WAX system originally used RSA for digital signatures. However, problems cropped up. In particular, RSA digital signatures require a Public Key Infrastructure (PKI), expiring keys cause problems for long-lasting information, compromised keys are difficult to address, and RSA-DSI royalties were expensive. As a result, WAX uses one-time signatures as an intermediate solution.

New HTML elements allow hashes and public keys to be embedded in documents. In addition to the standard linking information, the A element also includes a HASHVALUE parameter. When a browser follows a link, it can hash the appropriate contents and verify whether the document is authentic. For instance, a link may appear as `<a HREF="http://www.med.ac.uk/examresults" HASHMETHOD="TIGER" HASHVALUE="12345..." HASHPARENT="http://www.cert.med.ac.uk">link</a>`. The exam results page would contain further information to reconstruct the hash.

Pure ERLs apply easily to static texts (e.g., health care, law and contracting, banking). One can also store hashes with

bookmarks for change control. Additionally, this system can interact with public key mechanisms. Work progresses on medical applications (WAX, British National Formulary), incorporation of XML discussed with industrial partners, and formalization of the ERL logic extended by public key parameters.

For more information, email vm206@cl.cam.ac.uk or visit <http://www.cl.cam.ac.uk/~fapp2/papers/ec98-erl/> and <http://www.medinfo.cam.ac.uk/wax/>.

Detecting Hit Shaving in Click-Through Payment Schemes

Michael Reiter, AT&T Labs - Research; Vinod Anupam and Alain Mayer, Lucent Technologies, Bell Laboratories

Mike Reiter, presenting the winner of the workshop's Best Paper award, analyzed several mechanisms to calculate upper and lower bounds on referrals to another Web site. This is particularly useful in Web advertising schemes where a Web publisher receives a payment directly proportional to the number of "click-throughs" generated.

A user U "clicks through" site A to site B if A serves a page to U, and then U clicks on a link in A's page to reach B. Here A is the referrer and B is the target. In a click-through payment scheme, B pays A for each referral that A gives to B. There are two common forms of fraud in click-through payment schemes. Hit shaving results when site B fails to credit site A for referrals. Hit inflation results when site A causes bogus referrals to site B.

Reiter described two classes of practical and immediately useful techniques for detecting hit shaving. In a heuristic approach, the target site need not cooperate or even have knowledge of the process. In a cooperative approach, one can achieve better accuracy and non-repudiation of click-throughs. For both classes, the detection techniques are mostly invisible to the user.

The detection process must enable site A to monitor how often site B receives a request from any user U with a referrer field indicating A. This leads to the question of how to calculate upper and lower bounds on hit counts. Site A can record an upper bound on its referrals to site B with no cooperation from B. When user U clicks on a link to site B, A is told about the click. Then user U continues to B. One can implement this using HTTP redirection or a CGI script. A second approach uses JavaScript and an invisible frame to notify site A of the intent to follow a link. These techniques produce an upper bound because one cannot be sure whether B actually receives the hit. The notification represents the intention to visit site B, but not a guarantee to visit site B.

Techniques to calculate a lower bound are not so clean or simple. After a user follows the link on site A to reach site B, the user notifies site A. A receives notification only if the connection to B worked. Reiter described a rather complicated procedure which spawned a new browser window and used JavaScript. Since one window cannot access another window's namespace, there are a few hoops to jump through. A detection window probes the namespace of the window attempting to contact site B. When the detection window is no longer allowed to probe the other window, it knows the connection to site B was successful. The detection window then notifies site A by requesting a particular URL.

The lower bound technique has a few caveats. The user might close the window before A is notified. Additionally, this only detects that some page is loaded. The user may have stopped the request to site B and traveled elsewhere. A few tricks (e.g., hiding the toolbar) can make it hard for the user to bypass the notification process, but it also can cause annoyances to the user.

Reiter suggests using both lower and upper bound detection on referrals. The

two measurements should be fairly similar.

In the cooperative approaches, site B acknowledges each referral as the referral happens. In a naive solution, B would open a connection to A for each hit. In a distributed approach, B's page would make the user request another page from site A as an acknowledgment. It is also possible to provide for nonrepudiation with digital signatures. B includes a digital signature while serving a page. However, this could easily become prohibitively costly. Hash chaining can alleviate some of the performance problems.

Reiter revealed a few disadvantages of hit shaving detection. There is a negative impact on user privacy. Web sites can discover your browsing habits. The schemes are also incompatible with anonymizing services such as Crowds or LPWA.

Questions began on a humorous note. How did Reiter become involved with this project? The saga began when Reiter placed his email address on a Web page. A spammer sent an email about click-through payments saying that a 1998 Corvette would be awarded for the highest number of click-throughs. Thinking something must be fishy, Reiter began to analyze click-through payment schemes. A few questions about ethics and morality popped up. All concerned impediments to the user (e.g., awkward windows popping up) and pornography. Reiter cleverly escaped the questions with witty remarks. However, he made it clear that improving the porn industry is not his goal. Click-through payment schemes are relevant for all types of Web advertising. Finally one attendee pointed out that these schemes act like a poor man's Remote Procedure Call via URLs. Asked whether he was on to something bigger, Reiter replied that there might be overlap or some related opportunities.

Session: Consumer Service

Summary by Matthew Hohlfeld

Sales Promotions on the Internet

Manoj Kumar, Anand Rangachari, Anant Jhingran and Rakesh Mohan, IBM T.J. Watson Research Center

Manoj Kumar presented his group's work on translating another set of real-world practices to the world of electronic commerce. Their specific goal is to provide the types of sales promotions that are not currently available at commerce sites on the Internet, namely coupons.

Kumar gave a list of fictional examples to illustrate the variety of purposes that can be addressed by coupons, and introduced the concept of eCoupons. The Internet disrupts several real-world assumptions and also allows some things that are not possible in the real world. One of the economic forces created by coupons is price discrimination. Price discrimination does not follow the understood rules on the Internet, however, if coupon "trading" is allowed to be as simple as it would appear to be.

One unexpected feature of coupons in the real world that issuers rely on is that they are "hard" to use. If the straightforward methods for creating coupons are used in the setting of electronic commerce, with the expected simple user interface, that would no longer be true. Kumar surprised me by discussing similarities between eCoupons and digital cash systems with respect to susceptibility to fraud, without stating that eCoupons are a type of digital cash (with a more complex "value" associated).

eCoupons are a set of definitions that are being refined and extended to attempt to cover all possible variations on promotions desirable on the Internet. Kumar's group is also investigating different methods for delivery of coupons and methods for storage after delivery (such as digital cash "wallets").

Perry Metzger stated that he thought that the marketing analysis in the presentation was better than the technical analysis, and that affinity schemes seem to be doing most of what is presented as eCoupons; many of the ideas are similar, just broader in scope. Avi Rubin wondered why eCoupons are needed when immediate discounts have all of the needed features listed; the only response was that businesses seem to want to continue to use what they "know." Max Tsvetov suggested that the use of auctions and other variable or negotiated prices is common and handles most of these needs; the response was that they solve orthogonal problems.

General-purpose Digital Ticket Framework

Ko Fujimura and Yoshiaki Nakajima, NTT Information and Communication Systems Labs

Ko Fujimura presented a flexible digital ticket project whose main purpose is to develop a generic value-circulation medium that prevents double-spending. In this context, a ticket is a digital medium that guarantees certain rights to the owner of the ticket. Describing tickets this generally allows the tickets to contain many different values and types of values in a single ticket (or group thereof).

Fujimura claims that a general ticket framework will reduce the implementation cost in many cases because a single design can be used in many places. By being general, the tickets can be composed arbitrarily, allowing bundling and similar features. He claims that the creation of new businesses to run this framework, like issuing/revocation services and deposit box services, was a benefit.

A general-purpose digital ticket framework must meet most of the requirements of digital cash. Additional requirements are: (1) A ticket can control its anonymity, divisibility, and transferability depending on the application; (2) The

individual specifications of a ticket need to be "machine understandable" to allow for the redemption of goods or services; (3) Ticket properties whose values change while it is circulated (e.g., payment or reservation status) must be changed securely; (4) A ticket that comprises more than one sub-tickets must be supported.

To implement such a framework, the authors created a Ticket Definition Language that allows for the specification of ticket properties. The tickets themselves are hypertext-based, allowing automation of the state-transitions and composability features. The tickets can also include dynamic information that is up-to-date when the ticket itself is used. Another feature (of less obvious utility) is that the tickets can contain very large data such as images and sounds.

The tickets themselves are inherently online (because of their hypertext basis and dynamic nature), but can also be circulated offline using smartcards. In either case, the system uses signed URIs to test the currency of the ticket. The meaning and constraints of the properties in the tickets are defined using the Resource Description Framework. Schemas for tickets can thus be controlled by the issuers of the tickets, and various restrictions can be contained in these schemas.

Fujimura outlined the ticket trust model. The issuer certificate, user certificate, and examiner certificate, which are required to issue, transfer, consume, or examine a ticket, are specified in the ticket itself using the Ticket Definition Language. So, any ticket with PK, such as drivers' licenses, can be used as a PK certificate if a ticket specifies them as a required certificate for the ticket. In other words, any ticket can play a roll in the PKI for other tickets.

They are drafting specifications for the implementation and intend to submit them to standards organizations. The goal of their project is to "Transform any Web terminal into a ticketing machine for any ticket in the world."

Paul Syverson spoke from the audience to indicate briefly that many related issues and solutions were addressed by Unlinkable Service Transactions.

Towards a Framework for Handling Disputes in Payment Systems

N. Asokan, Els Van Herreweghen, Michael Steiner, IBM Research Laboratory

Els Van Herreweghen spoke about dispute handling in a digital marketplace. While it is assumed in many designs that some offline dispute arbitration system exists, the designs do not include specifications for addressing this. Systems need to obtain evidence in order to resolve disputes, to show exactly what the evidence means, and to provide tools for the analysis of this evidence (inside and outside of arbitration). This evidence may be useful in situations other than litigation, such as local verification, friendly settlement, or showing a receipt to a third party. It should be possible to leverage the high-level interface of E-Commerce applications to address possible after-the-fact disputes. Van Herreweghen proceeded to give some clear examples what sorts of disputes these might be.

She introduced a language for dispute claims in first-order logic with who, what, modifiers, and attributes. All possible disputes in a given digital market system should be describable in this type of language. By representing protocols as a sequence of states where players in the protocol cause transitions between these states, she showed that a properly produced transcript of the protocol would provide evidence for claims. A claim verifier can take such evidence to determine which states the protocol reached, and use knowledge of what those states "mean" in the protocol to determine whether or not the claim is valid.

To provide proof that this method will work for automatic dispute handling, the authors are working toward building implementations for iKP and SET. They

suggest a three-party dispute resolution system in which the verifier interacts with the claimant and the respondent, accepting evidence and making a decision based upon the claim and the evidence. Van Herreweghen admitted that different scenarios will require the use of different parties in the arbitration, and gave a brief overview of how to add dispute handling capabilities to a given payment system.

Max Tsvetovat wanted to know about the applicability of this work to level commitment contract systems (contractual protocols); Van Herreweghen answered that if the claims and contracts are expressed correctly, these techniques should work. Paul Syverson asked where the formalism that is used originated from; the answer was that the authors were inspired by generic payment system interfaces, but there were no existing generic payment system-independent claim languages, so they developed a new language for payment disputes. Syverson also wanted to know about the overlap between this and other provable logic systems; the answer was that they want a usable system, and an unused/unusable formulation. (According to a later discussion with the speaker, the focus of their work has been mainly on the generic claim language and framework for payment disputes. They were concerned primarily about the usability of their approach to existing, nonideal payment systems. They are involved in research into using other approaches and other logics for dispute resolution systems.)

Session: Short Talks/Works-in-Progress Reports (WIPs)

Summary by Matthew Hohlfeld

Robert Hettinga is working on a full flow diagram of commerce in “Cypherspace” (or encrypted Cyberspace). He claimed that one of the most important features is a peer-to-peer design, allowing anyone in the system to play the role of “merchant” or “purchaser.” A white paper and the completed diagram will be forthcoming and available on his Web site.

John de Pre Gauntt presented a non-US-central point of view on electronic commerce. He pointed out that market penetration of mobile phones is much higher than that of personal computers in other parts of the world, and proceeded to show how they could be (and are being) used as a vehicle for electronic commerce.

Vinod Anupam suggested that even though there are many well-known attacks against the implementation of JavaScript in currently available commercial browsers, it is possible to fix this problem in coming versions. His group has been working on an implementation that uses a Safe Interpreter with controllable security policies by building on the source code provided for Mozilla. They are attempting to get this feature rolled into the official Mozilla 5.0 when it is released.

Otto Koppius presented the audience with a possible view of how multidimensional auctions could work. He noted that the auction itself will be dependent upon the

evaluation function of the individual selling the item. Koppius is currently involved in examining what will occur with different scenarios that are made possible by multidimensional auctions.

Max Tsvetovat is looking at the issues involved in using agents to negotiate and execute contracts. The situations that his group is considering include multiple-supplier, single-customer arrangements. They are looking at a “one-shot, leveled-commitment” protocol to perform the negotiation, as well as other new topics.

Bob Carter described his company’s efforts in deploying smartcards in a closed public key infrastructure in the context of delivering smartcards to account holders for a bank.

SAGE news & features

Coming up to Speed



by Tina Darmohray

Tina Darmohray, editor of SAGE News & Features, is a consultant in the area of Internet firewalls and network connections, and frequently gives tutorials on those subjects. She was a founding member of SAGE.

<tm�@usernix.org>

I've been a UNIX system administrator for 15 years. That number isn't really accurate, however. I've been employed as a system administrator that long, but it's probably a stretch to claim that I've been a fully functional sysadmin the entire time. Like so many system administrators, I learned just about everything I know about system administration while on the job. So, while I was first hired in 1983, I probably wasn't worth a significant part of my salary for a good bit of time after that.

Luckily, I wasn't alone. Many of the employees on the project were also learning on the job. In fact, a small number of the more senior staff members readily admitted that they were there solely to learn about UNIX. They had come from years in the mainframe world and wanted to get some experience with the "new" operating system. They'd been working for years by then, and it showed. They were already senior system administrators and programmers. They knew about compilers, revision control systems, report generators, network stacks, system backups, IP addresses, disk drives, and all the rest of the standard computer fare. For the most part, they weren't learning new concepts; they were just applying what they already knew to a different environment and learning how to "do it in UNIX." They were coming up to speed with the "new" operating system.

Now I find myself in a similar situation. While I'll possibly never approach the competence of some of the senior staff members on that project, I'm no longer just learning my trade. I have a lot of experience to draw on now. Just as they were then, given a completely new OS environment, I'm still much farther ahead than someone just starting with his or her first one. I'm coming up to speed with NT.

Since it worked last time, I'm approaching this learning curve in the same way I did with UNIX. I've got some machines to play with, I've bought books, I'm reading articles and online information, and I'm talking with peers. In general, I like to learn, so I've been enjoying the opportunity to really delve into something new. Sadly, it all came to an abrupt stop a few weeks ago when I was pretty far down the path to understanding the NT authentication process and then hit a barrier. It was a Microsoft proprietary algorithm. Here I am, trying to educate myself so I can get better at NT, and I get to a point where I can't have an answer! It prompted me to think about how one "gets good at" something, which took me back to my first system administration job and how we all learned about UNIX. With UNIX, if you want to know something, you can buy books, talk to coworkers, read USENET (prior to the Web), attend tutorials, and, if you have the determination, you can finally resort to printing out the source code and figuring it out! Granted, you may wind up learning more than you ever needed to know, but ultimately that's your call, not someone else's, to make.

Somewhat deflated, I wondered if it was "just me." So, over the past several weeks, I've been polling people that I talk to see if they've had a similar experience or perception with NT. To a person, they have. Not all of them were bothered by it, but all of them agreed that, in some respects, your hands are tied from becoming an "expert" (in the traditional source-code-level "UNIX-guru" sense) with NT.

Remarkably, I also ran into a related business article that said that some large companies were, surprisingly, choosing Linux over NT, even though it's not supported. Apparently, the reason is because companies want to be assured they have control over fixes, which isn't guaranteed when Microsoft "owns" the operating system. That makes sense to me, too, but I wouldn't have predicted it.

So, I'm left to wonder what this means for the level of expertise of NT system administrators. On one hand, my elementary school kids can use the popular word processing package on NT, almost intuitively (and, realistically, that's never going to happen with "vi"). From a system/network administrator perspective, however, I'm feeling a little like I'm left at the GUI level too. The high-level, and sometimes proprietary, NT interface to the system may mean that the NT "gurus" of the future have to settle for a more abstract level of knowledge than the UNIX gurus. I suppose we've lived with this knowledge restriction before, as in the days of OSes like MVS and VMS, but I'm not sure it's as much fun.

End of Year Report

by Hal Miller

Hal Miller is president of the SAGE STG Executive Committee.

<halm@usenix.org>

It's been more than a year since I gave what probably should become an annual status report, so here is the State of SAGE, a summary of what we have been doing over the last couple of years.

SAGE once operated on a series of "understandings." That worked reasonably well as long as the same people stayed on in all the relevant positions. Once those people moved on, the corporate history left with them, and those who followed spent a considerable amount of time trying to recreate the "rules" under which SAGE operates. To fix this permanently, we wrote SAGE's first set of "articles of organization" and an accompanying policy documents. Included in these are a rewritten charter, a mission statement, and various changes such as the synchronization of elections.

USENIX and SAGE hired a Webmaster. The SAGE Web site has grown tremendously, with lots of new items that go toward fulfilling the vision of SAGE. Some examples are the Code of Ethics, Speakers Bureau, ToolKit, Rosetta Stone,

How-To Notes, Web Resources for NT.

An editor was hired for the Short Topics Series. That series was itself rather short: there was only one entry when I was first elected to the executive committee. By the time you read this you should have four booklets on your shelf, and a fifth is due shortly. Others are in the pipeline.

In addition to an ever-growing LISA, we now co-sponsor LISA-NT and the new network admins conference. This year's LISA includes a new workshop entitled "globalLISA," as LISA continues branching out to explore more problem areas faced by our members. We are investigating similar expansion for Samba and other areas.

We conducted surveys to see what you want from SAGE (and found exactly the diversity of opinion we expected). One area the surveying pointed out was your desired involvement in the standards process, so under Nick Stoughton's watchful eyes I gave a two-hour talk to a program committee of The Open Group about areas in which standards could help working sysadmins. We are looking at ways of continuing representation to this committee.

Together with USENIX, we initiated an area we call "Good Works," in which a Virtual High School session in Maryland was sponsored to teach students the basics of system administration. We are

now looking at ways to make expanded use of the resulting curriculum, as well as at additional projects in a similar vein.

Our membership numbers, and our contacts with sysadmin groups outside the US continue to grow. There are a bunch of new local groups, who have SAGE-sponsored speakers from time to time.

We have an email-based pilot "mentoring" program in which one can go for individual question help (the "SAGE Oracle," coming soon.) And then there's the "Day In The Life" survey and a SAGE History compilation.

So what next? Here are four questions this organization needs to contemplate as we move forward.

What should SAGE be? This seems to me the real key question for the future. According to our charter, vision, and mission statement (see <<http://www.usenix.org/sage/official/official.html>>) SAGE is a "professional guild." It is "a membership group of system administrators organized to further, foster and recognize the profession." The role transcends operating system types, vendors, and locations. The purpose is to provide benefit to system administrators in their training, workplace environment, career advancement, and as a "place" a member can go for assistance. We are a primary point of contact for vendors – who else represents the people they actually want to sell to?

SAGE, the System Administrators Guild, is a Special Technical Group within USENIX. It is organized to advance the status of computer system administration as a profession, establish standards of professional excellence and recognize those who attain them, develop guidelines for improving the technical and managerial capabilities of members of the profession, and promote activities that advance the state of the art or the community.

To achieve its mission SAGE may:

Sponsor technical conferences and workshops;

Publish a newsletter, and/or professional short topics series;

Develop curriculum recommendations and support education endeavors;

Develop a process for the certification of professional system administrators;

Recognize system administrators who are outstanding or are otherwise deserving of recognition for service to the professional community;

Speak for the concerns of members to the media and make public statements on issues related to system administration;

Promote and support the creation and activities of regional or local professional system administrators.

SAGE STG EXECUTIVE COMMITTEE

President:

Hal Miller <halm@usenix.org>

Secretary:

Tim Gassaway <gassaway@usenix.org>

Treasurer:

Barb Dijker <barb@usenix.org>

Members:

Helen Harrison <helen@usenix.org>

Amy Kreiling <amy@usenix.org>

Kim Trudel <kim@usenix.org>

Pat Wilson <paw@usenix.org>

We are neither a union, a guild, a trade, or a craft. We are not likely ever to be an industry-controlling agency like the national medical and legal associations, although we gain insight from parts of their positions and methods. We are a loose organization of diverse folks who have much in common, and are mutually supportive.

In view of this, *what should SAGE look like?* Not very different from what we already are, but there are some adjustments I'd like to see. SAGE in the US is a solid, alive, functioning organization. We have a central national directive body, financial backing, local groups, and a set of rules to play by. We still have problems in two areas: delegation and internationalization.

As with any volunteer group, we have a problem with delegation. Most of the work at any one time is done by a few people, despite lots of interest, lots of good intentions, and lots of promises to accomplish specific goals. SAGE has made a number of attempts to delegate work to more people, but we're still floundering around trying to find a method that will work. Lots of us sign on for specific tasks in the enthusiasm of the moment only to find that our families and hectic work lives take precedence over volunteer work. Thus, most of the work of keeping SAGE going falls on the

shoulders of a minority of the executive committee. Somehow, we need to involve more of our membership in the work we all think needs doing.

We also need to remember that we need to look beyond our American system administration community, and coordinate better with our colleagues abroad. We all have the same problems, solutions, and needs, so we should work together.

Thirdly, *what should SAGE be doing?* Again, I think we're on track for the most part, although with a long way to go. Looking back at our purposes, it seems to me that the to do list should include: education, certification, expanding beyond the UNIX box, speaking for the community (which includes involvement in the standards process, representing sysadmins to the vendor community, and working with sysadmin management), and expanding the technical advancements in our field.

Our educational needs include work in the "formal" and "informal" areas (see my December, 1997, *login:* article) and developing a continuing education plan both within and outside of the certification arena. We need to expand downward into high schools, as well as to people who are thrust into parts of our job from other (often nontechnical) backgrounds.

We are doing better at becoming less

UNIX-focused, but we need to ensure we don't move from "one OS" to "two OSs" and instead generalize to "any OS" – applying rules of scale. We have been talking recently in the executive committee meetings about closer ties with vendors, but haven't yet worked out any plans. This is something for progress during the next couple of years.

Our current standards involvement is pretty minimal: we've asked Nicholas Stoughton, the USENIX Standards Liaison, to add our interests to his repertoire. He's done that, but we need to follow through with some work on our part now.

Other than the booklets on job descriptions and security for managers, we've not done much for our employers/managers yet, and need to improve our own lot by improving theirs.

Finally, there is another difficult item on the to do list: technical research and development. SAGE has a history, through the USENIX Association and general conferences, of creating new tools or methods and reporting on them to the community at large. As we've become more separate from our USENIX developer community and more sysadmin oriented, we have been losing sight of that function. Our time is so limited by the pressures of our jobs we have that we're often lost in the trees and can't see the

SAGE MEMBERSHIP

<office@usenix.org>

SAGE ONLINE SERVICES

Email server: <majordomo@usenix.org>

Web: <<http://www.usenix.org/sage/>>

SAGE SUPPORTING MEMBERS

Atlantic Systems Group

Collective Technologies

Compaq Computer Corporation

Deer Run Associates

D.E. Shaw & Co.

ESM Services, Inc.

Global Networking & Computing, Inc.

Microsoft Research

New Riders Press

O'Reilly & Associates

Remedy Corporation

SysAdmin Magazine

Taos Mountain

TransQuest Technologies, Inc.

UNIX Guru Universe

forest. We wait for someone else to do the development, or worse, wait for vendors to decide what we need, without even giving them a hint. We need, as an organization, to find a way to sponsor new development work specifically for sysadmins, and get it going.

The final question is "How do we get there?" It takes a lot of work, most of which must of necessity be by volunteers. Given the difficulties sysadmins have finding time to volunteer, careful coordination of the volunteer time we come up with is important. We need to get out there and beat the bushes for new members, ties with vendors, paper writers for conferences, tool writers, article writers for *login*, mentors, teachers, etc. We need people to help organize conferences, to help staff the SAGE booth at conferences, to respond with opinions when someone "flies" an idea. We need help in expanding the SAGE model to ensure that it properly covers N+1 operating systems. How we get there is through organization and coordination. It is our biggest and most important task ahead.

I'm a sysadmin. We are trained by our daily work life to dive in and "do" on our own, often without regard to the possibilities for teamwork. One of the hardest things I've had to do is to keep my mouth shut on the SAGE-AU board once I left that presidency, and allow the new officeholders to run things however they saw fit. Allowing others to have the freedom to do things their own way is hard, but is the key to delegation. This task is the one to which the next Executive Committee must first lend itself. Not only do we need the volunteers, but we need the ability to manage and coordinate the work done.

Elections for the Executive Committee are coming up. I can't overemphasize the importance of returning your ballot. We have around 4,000 members. Most current members of the committee received between 300 and 400 votes each, and you get one vote per position. That means that 10% of the membership elected us.

Our surveys received about the same percentage of response. We've driven SAGE forward on the basis of our personal agendas and comments we received from a small number of you, I hope in the direction most of you wanted us to go.

SAGE Certification: Q&A



by Barbara L. Dijker

Barbara Dijker is Treasurer of the SAGE STG Executive Committee.

<Barb.Dijker@labyrinth.com>

Editor's Note: LISA will have a "Great Debate" about SAGE's certification effort. I have asked Barb to supply some information and will follow up with an interview with her next month. RK

Q. What are the benefits of certification?

A. Certification has long been used in other fields and is fast becoming typical in the computing field. Certification provides:

- an objective means to evaluate prospective employees,
- a means for system administrators to evaluate their own skills, and
- a basis for educational curricula.

Q. How is certification different from education?

A. The goal of education is to impart skills and/or knowledge. The goal of certification is to evaluate whether one has successfully attained such skills and/or knowledge. The two go hand-in-hand.

Commercial educational programs typically do not include any rigorous testing. If you paid for and attend a class, you "passed." Certification is the missing evaluation component.

Formal higher education involves evaluations for pass or fail. However, it tends to be more broad and much less specialized than is required to satisfy quickly fluctuating industry demands.

Q. Why is SAGE developing a certification program?

A. SAGE in the US and SAGE-AU are the only organized associations of system administrators. The best and most effective certifications programs are those specified by the people who practice in that field.

That is not sufficient motivation alone. The industry demand for system administrators has been growing for many years. The supply of qualified system administrators has not kept pace. This is in part evidenced by the growing number of commercial system administration certification programs. Certification is one way to mitigate this demand.

Certification is one way SAGE is supporting newcomers to system administration. Other projects include:

- fostering interest in computing starting in high school,
- supporting education through higher education, conferences, and publications,
- providing assistance to local groups, and
- developing a mentoring program.

Q. What are the goals of the SAGE certification effort?

A. The field of system administration, while being only a small part of the computing field, is quite diverse. For this reason, SAGE certification will initially focus on "core competency." As the program matures, certification may be extended to cover classifications or specializations, e.g., webmaster, postmaster, different operating systems, etc.

In defining and implementing a certification program, SAGE has two fundamental principles which it will uphold:

- the evaluations must be of merit: useful and meaningful to the participant and the community, and
- cost must not be a barrier to participation, i.e. purchase of books or courses will not be required.

Q. Who is involved in the certification development process?

A. The primary vehicle for the SAGE certification project is the Certification Subcommittee.

The subcommittee will make appropriate use of professional certification consultants. Just as system administrators are experts in their field, there are experts in the field of developing certification programs and methods of evaluation.

In addition, an Advisory Council has been established. The role of the Advisory Council is to provide a sounding board and feedback to the project.

Q. What is the time frame?

A. The three major phases to establish this certification program are:

1. Definition (skill requirements and program logistics)
2. Implementation
3. Production

The schedule calls for the definition phase to be completed by mid-1999 and the implementation phase to be completed by early 2000.

Q. Who will benefit from certification?

A. There are two intended beneficiaries of a SAGE certification program:

1. Budding system administrators who are interested in improving or defining their skills and/or having objective recognition of them.
2. Hiring managers of system administrators who might benefit from an external objective evaluation of applicant skills.

This potential certification program is not intended to license practicing experi-

enced system administrators.

Q. Why should I become certified?

A. When (if) the SAGE certification program is implemented, there are several reasons to become certified.

1. The certification program may be used as an educational goal. There are already several educational organizations interested in providing educational programs that impart the skills necessary to pass SAGE certification tests. Whether or not you actually take the tests, you may want to use educational courses and tools geared toward SAGE certification.
2. Certification, like any goal such as a college degree, provides an achievement to strive for. Since SAGE certification would be a single source evaluation, it would provide an objective assessment of your skills.
3. Also like other tangible goals, certification provides a recognizable result. Part of any certification program is recognition. Those who become certified may do so in order to gain the recognition promoted by the program. This recognition could result in increased job opportunities and/or better tangible job credibility such as salary.

Q. How may complex skills be tested?

A. This question will be answered during the development of the program. Whatever testing methodologies are used, they have to meet the program goals. Recall those goals require that the testing be "of merit." Therefore, any testing method which allows for rote memorization is not acceptable. There are professionals who specialize in the development of testing methodologies. Such professionals will be consulted to ensure an effective result. For example, testing for nursing or police work involves components that evaluate actual skills rather than "book knowledge."

Are You Preparing for the New Millennium?

by **Tim Gassaway**

Tim is a member of the SAGE Executive Committee.

<gassaway@usenix.org>

The end of the year is upon us already and it's time to start thinking about 1999. That's right – we are only about 365 days from the new millennium. With all the rage over Y2K issues, system administrators around the globe will be challenged more than ever to be the ultimate problem solvers. Therefore, it's vital that you have an organization like SAGE around to support you at this critical time.

SAGE members can support one another by participating in a moderated exchange of information on current and future issues that are important to system administrators. If you're not already subscribed to <sage-members@usenix.org>, now is the time.

Keeping that in mind, I would like to dub the year 1999 as the year of the volunteer. Remember the famous quote from JFK, "Ask not what your country can do for you – ask what can you do for your country." Well just substitute "SAGE" for "country" and I think you'll get the message. I would like to challenge each member of SAGE to volunteer just a small fraction of time to assist with a SAGE initiative or event.

What initiative or event? Well, I have listed a few below and invite you to visit the <<http://www.usenix.org/sage>> for additional initiatives and events.

The 1st Conference on Network Administration, scheduled for April 7-9, 1999 <<http://www.usenix.org/events/neta99>> will provide a wealth of information pertinent to network and system administrators. The 2nd Large Installation System Administration of Windows NT Conference is scheduled for July 14-16,

1999, in Seattle, Washington. There is still time to submit proposals before the deadline on February 23, 1999. The 13th Systems Administration Conference (LISA '99) is the last one before the year 2000. This conference is scheduled for November 7-12, 1999, in Seattle. We are going to have a great time finding out about all those last minute Y2K issues. Now is the time to start planning to participate in the conference by submitting a proposal.

In addition, the SAGE series editor has issued a request for proposals for the Short Topics in System Administration series. The topics are:

- Effective Customer Support
- Monitoring Techniques and Practices
- The Role of Postmaster
- The Role of Webmaster

The list of opportunities is endless. So here's the deal: send me an email <Subject: SAGE Volunteer> listing what your interests, skills, and availability are, and we will put you to work. I'll be expecting a full mailbox soon.

Making Users Happy

By Sean Kamath

Sean Kamath is the manager of the CPID Engineering System Administration (CESA) Team. CPID is the Color Printing and Imaging Division of Tektronix.

<kamath@pogo.WV.TEK.COM>

"If users are made to understand that the system administrator's job is to make computers run, and not to make them happy, they can, in fact, be made happy most of the time. If users are allowed to believe that the system administrator's job is to make them happy, they can, in fact, never be made happy. Furthermore, in their quest for happiness, they will cause enough resources to be diverted to trying to make them happy that the computers will no longer run."

— Paul Evans (as quoted by Barb Dijker in "Managing Support Staff," LISA '97)

A couple of days ago I started a mini email flame-fest on the SAGE mailing list. I responded to a comment about making users "happy." My response was, basically, "Who cares if the users are happy, what matters is making them productive." (Of course, I really do care, but I was cranky that day.) Needless to say, there were a number of responses about how keeping the users happy was the most important thing they do, particularly from the ISP community. I've been spending a couple of days thinking about this, and I still don't agree. The focus of system administration is Productivity, not Happiness.

Throughout this article, I'm going to use the word "productive" a lot, but keep in mind I really mean "able to get something done in an acceptable time-frame with acceptable costs." "Productive" is just shorter.

I understand that there are significant differences among corporate, ISP, government, educational, and other types of system administration. However, the title

"system administrator" has an implication that transcends these differences. We "administer systems," not "make people happy" or "entertain people" (at least, generally, not intentionally). The word "system" in this case really means more than just the hardware and software on the machines we run; it includes networks and people. Our goal, first and foremost, is to make sure that the systems we administer are useful and productive. Otherwise, why have the system to begin with? Having machines sitting around in dark rooms with no one using them (or their output) is not very productive for anyone.

This raises the question "Who should be responsible for determining if the users are productive?" Depending on functions and/or organizations, different people or groups of people have this responsibility. In corporate shops, it is usually the manager of the end user who determines productivity. In educational areas, it might be the professors. For ISPs, clearly it's the end user. I'm not sure who it is in government, probably appropriations committees.

What it boils down to is that our job is to help people get things done. The end result is usually a satisfied customer, one way or the other. The important difference between targeting end-user productivity instead of happiness is that the former is measurable (albeit often not by the sysadmin directly), objective, and visible to others in a nonemotional way. You also have a better chance of justifying decisions on the basis of increasing productivity, rather than making users happy.

Often the end users are not fully able to determine what will make them happy – or even productive – especially in the long term. For example, the end user might demand that you stripe a set of disks. But they might have been happier to have a RAID5 or RAID 0+1 array after the disk died and had to suffer a 24-hour restore, assuming there was enough disk

space. The balance here is to meet their needs, both short- and long-term, in terms of productivity.

By the way, I'm not advocating the traditional RTFM, since in many cases one cannot expect an end user to read documentation and get out of it what a sysadmin can. Yes, we would all like people to help themselves first, before coming to us for further assistance, but the fact is that's not how things work. How would you feel if your doctor said, "Read the manual" every time you went for an office visit? Doctors are paid for their experience and knowledge, just as we are paid for ours. It's possible that end users can figure out stuff on their own from documentation, but it's more productive (for them) to get a quick answer from someone who already knows.

There is also a tremendous sense of what I call *perceived productivity*: since an end user has all the equipment he or she needs, that user will automatically be productive. But if you put the world's fastest UNIX box on a Windows-only user's desk, it doesn't matter how fast the UNIX box is; they won't know how to use it. Don't fall into the trap of believing that if we give an end user everything we would need to get the job done quickly, that the user has everything he or she needs to get the job done quickly.

A co-worker I was talking to about this topic said that making users happy sometimes is more important than productivity. He cited a major Internet provider that has a lot of GUI-based stuff, but not a lot of substance behind it, implying that its focus is happy users instead of productive ones. My response was that their goal was to get as many people productive (i.e., getting something out of the service) as they could, and that his perception of what would make the users productive didn't matter. It's the users of the ISP that decide if they are getting their Web pages and other services for which they are paying.

By facilitating a user's productivity, you tend to make that user "happy."

Therefore, you should focus on productivity as the goal. Regardless of how you choose to view end users (customers, pests with requests or losers, team members, or coworkers) the point is that we were hired (by someone) to help them get something done. Tom Limoncelli's excellent paper at LISA '97 ("Turning the Corner: Upgrading Yourself from 'System Clerk' to 'System Advocate'") has an excellent section on your attitude as a system administrator. And remember: In one way or another, we're all end users, and we all have something we want to get done. Making users productive – not making them happy – should be your primary focus.

Making your spouse happy, now that's another story.

Report of the Nominating Committee

By Pat Wilson, Amy Kreiling, and Greg Rose

The Nominating Committee is pleased to present a strong slate of nominees for upcoming SAGE election. Each of the following candidates has indicated her/his willingness to serve on the SAGE Executive Committee for the 1999-2000 term.

The Committee nominates the following individuals:

- Bryan McDonald, Global Networking and Computing, Inc
- Hal Miller, University of Washington
- David Parter, University of Wisconsin
- Peg Schafer, Harvard University
- Bruce Wynn, Collective Technologies

The election for the SAGE Executive Committee will be held in December, 1998, with the newly elected committee members taking office no later than February 1, 1999.

SAGE members will have an opportunity to meet the candidates at the upcoming LISA '98 conference in Boston; there's a Candidates's Forum scheduled at the conference on Thursday, December 10, from 5:30-6:30pm. Please come, then vote responsibly.

- Barb Dijker, Labyrinth Computer Services
- Tim Gassaway, Auspex
- Xev Gittler, Goldman, Sachs & Co.
- Geoff Halprin, The SysAdmin Group
- Jim Hickstein, consultant



**by Elizabeth
Zwicky**

Elizabeth is technical lead of the European Desktop Project at Silicon Graphics. She was a founding member of SAGE and is currently on the USENIX Board of Directors.

<zwicky@pterodactyl.neu.sgi.com>

Enough SNMP to Be Dangerous Part 1

This is the first of a series of articles dedicated to teaching typical UNIX system administrator types – people who can compile public domain software and have some idea about TCP/IP – how to do UNIX-style hackery with SNMP. It is not an elegant and systematic approach to SNMP, but it should give you enough background to be dangerous.

SNMP, the simple network management protocol, is one of those simple but powerful (and therefore brain-bendingly complicated) concepts. The simple part of SNMP is that it basically supports only two commands and a few variations on them. The commands are `get`, to get the value of a variable, and `set`, to set the value of a variable. The hard part is figuring out (1) what variable you want and (2) what the value means.

Because it doesn't require vast intelligence to speak SNMP, all sorts of things speak it. Among the devices with SNMP running on my local-area network, where we weren't using SNMP and therefore hadn't done anything special to turn it on, are almost every UNIX box, every router or switch with an IP address, every printer with an IP address, every NT machine or NT-variant machine, a significant number of machines running Windows 95, and the voicemail system.

This means that a person armed with some freely available tools, some basic SNMP knowledge, and some interest in poking at network devices with blunt sticks, can come up with all sorts of interesting and occasionally even useful information.

Freely Available Tools

The first thing you need, of course, is the tools. The pre-eminent tool suite for SNMP is from the University of California at Davis, and is available from <http://www.ece.ucdavis.edu/ucd-snmp>. On top of that, you will probably want a Perl library; I use G. S. Marzot's. (I have to admit, I didn't do a comprehensive survey, I just picked the first one I ran across that worked): <ftp://ftp.wellfleet.com/netman/snmp/perl5/SNMP.tar.gz>.

Basic SNMP Knowledge

One of the not-so-simple things about SNMP is that it has its own terminology. This is not entirely unreasonable as a way of keeping you from making assumptions, but it can be daunting. At base, SNMP has two parts. One of them is a server that sits on the device; this is usually called an "agent." The other one is a client that asks questions; this is usually called a "manager."

The main reason these things are not just called clients and servers is that the server half may be capable of initiating messages itself (called "traps"), in which case something that's normally a client needs to be sitting around listening for them. In this situation, the client-server paradigm is bent rather badly.

The objects that the agent and the manager exchange are defined by something called a MIB (Management Information Base). A MIB specifies a mapping between long dotted numbers, human readable names, and pieces of information. For instance, .1.3.6.1.2.1.1.1.0 is more familiarly known as `system.sysDescr.0`, and it contains a description of the system. This is what the MIB that says this looks like:


```

RFC1213-MIB DEFINITIONS ::= BEGIN

IMPORTS
    mgmt, NetworkAddress, IpAddress, Counter, Gauge, TimeTicks
    FROM RFC1155-SMI
    OBJECT-TYPE
    FROM RFC-1212

mib-2 OBJECT IDENTIFIER ::= { mgmt 1 }
system OBJECT IDENTIFIER ::= { mib-2 1 }

sysDescr OBJECT-TYPE
    SYNTAX DisplayString (SIZE (0..255))
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "A textual description of the entity. This value should
        include the full name and version identification of the sys-
        tem's hardware type, software operating-system, and networking
        software. It is mandatory that this only contain printable
        ASCII characters."

 ::= { system 1 }

```

Most agents implement at least two MIBs, a standard base MIB and one specific to the device. Many of them implement more than that. A manager that talks to many different agents will quickly end up using dozens of different MIBs. That's okay, because the manager software we're using is easygoing in several ways. First, it doesn't really care if it has a MIB or not; if you're willing to use numbers or "get next" to get unparsed information, the software will gladly forgo the work of translating things into names. Second, it will read text form MIBs. Fancier management software generally wants to compile MIBs into more efficient forms; with the UCD package, you can just FTP over text files and shove them into its directory. I'll talk about how to find MIBs later; UCD provides the most basic MIBs along with the software.

The final term you need to understand is "community string." The community string is a token passed from the manager to the agent; you might want to think of it as a remarkably weak password (it is passed around in cleartext). The device you're talking to will use the community name you give it to decide what data you should have access to. The default community is "public," which should theoretically give access only to "safe" information. In practice, vendors have an unfortunate tendency to allow *all* SNMP to the community "public"; this may include the ability to get information you might not want given out to anybody in the universe, like the names of all the accounts on your machine, or worse yet it may include the ability to do sets on arbitrary variables. Since SNMP is used for device management, and it has only two commands, *set* does a number of things that you might not expect; if you're thinking "What harm can setting a variable do?" consider the possibility that it's the "Reboot now" variable, and has just been set to "true." Obviously, this sort of approach provides the ability to do all sorts of truly moronic SNMP tricks, most of which will not be discussed here.

Poking at Network Devices with Blunt Sticks

Once you have installed the UCD tools, pick a host that you're reasonably certain is running SNMP, and try this:

```
snmpget hostname public system.sysDescr.0
```

which will return something like

```
system.sysDescr.0 = Silicon Graphics Challenge/1 running IRIX 6.3
```

In practice, vendors have an unfortunate tendency to allow all SNMP to the community "public"; this may include the ability to get information you might not want given out to anybody in the universe, like the names of all the accounts on your machine, or worse yet it may include the ability to do sets on arbitrary variables.

*For a stupid SNMP trick,
try asking all your
local systems for
system.sysDescr.0 and
see how many of them
return something that
complies to the RFC
definition.*

if it works, and something like

```
No Response from hostname
```

if it doesn't.

Obviously, the first parameter to `snmpget` is the name of the host you want to talk to. The second is the community string. If you, or your network managers, already know about SNMP, "public" may not be an appropriate community string, or may give you only tame information suitable for the sort of tricks I intend to talk about. If you get "No Response" from a device that you are sure should be speaking SNMP, try asking somebody if you need to use another community string.

The second parameter, `system.sysDescr.0`, is the variable name that you want the value of. `system.sysDescr.0` happens to be the first value defined by the absolutely most common MIB for SNMP devices to implement, which is usually called MIB-II, is defined in RFC1213, and is provided with the UCD SNMP libraries. `sysDescr` also happens to be both human-comprehensible and relatively amusing.

For a first stupid SNMP trick, try asking all your local systems for `system.sysDescr.0` and see how many of them return something that complies to the RFC definition above.

Here's a sample of my results (nonprintable characters have been converted to ASCII equivalents).

Fails the mandatory requirement to be printable:

```
NX-500      ,2.0.6E^@  
CyberSWITCH-100 (ISDN v5.2) v2.0^M  
Copyright (c) 1996 Cabletron Systems, Inc.^M  
Copyright (c) 1995-1996 FlowPoint Corp.^M  
Copyright (c) 1985-1996 MPX Data Systems, Inc.^M  
All Rights Reserved
```

(What *are* they copyrighting here? Is it legal for me to publish this?)

Entirely printable, but missing either hardware or software. Note that Cisco manages to list software only on one machine, and hardware only on another.

```
Linux version 2.0.32 (root@linux95.corp.sgi.com) (gcc version  
2.7.2.1) #1 Tue Dec 16 21:34:57 PST 1997  
Microsoft Corp. Chicago Beta  
Cisco Systems Catalyst 1900  
Cisco Internetwork Operating System Software  
IOS (tm) 4500 Software (C4500-J-M), Version 11.1(7),  
RELEASE SOFTWARE (fc2)  
Copyright (c) 1986-1996 by cisco Systems, Inc.  
Compiled Wed 23-Oct-96 21:16 by tej
```

Not interpretable by nonexperts:

```
X8523L-3.4.7  
ES/1 ATX  
ATX Release 3.1.11 24-Jul-96  
PPE 512-0078-003 B 0002E400E2C5  
IF512-0054-003 X10002E400AE76  
QE512-0069-00B0002E4005FBC  
QE512-0069-00B0002E400C284
```


And just to prove that it can be done, a variety of vendors with their own eccentric formatting, but the right information (note that I'm willing to live without the network version, which only some of them include – a more draconian reading of the spec would pass only the last three).

```
Hardware: x86 Family 6 Model 1 Stepping 9 AT/AT COMPATIBLE -
Software: Windows NT Version 3.51 (Build Number: 179
        Multiprocessor Free )
SGICOR - OCTEL OVERTURE 300 Voice Mail System, Id: 302164 Rev 2.0.1
Silicon Graphics Challenge/4 running IRIX64 6.2
SunOS cambio 5.5.1 Generic sun4m
Tektronix, Inc., Phaser 360, PhaserShare Series B Network Interface,
(5.44/1.62/9.16)
Model; LANplex 2500, h/w rev; 06-0D, s/w rev; 04-03-00-07
80486 DOS 6.20
Windows 3.10 Enhanced Mode
NetManage SNMP 4.50
IBM RISC System/6000
Machine Type: 0x0030 Processor id: 000019913000
The Base Operating System AIX version: 03.02.0000.0000
TCPIP Applications version: 03.02.0000.0000
HP3000 SERIES 948, MPE XL version B.40.00 NS Transport version
B.05.00
```

I also ran into one machine that was running SNMP but did not have `system.sysDescr`; this is probably a symptom of gross misconfiguration.

Finally, here's the world's stupidest Web-based SNMP tool. It consists of a Web page with a form:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<HTML VERSION="2.0">
<HEAD>
<TITLE>Web Front End for SNMP Queries</TITLE>
</HEAD>
<BODY>
<H1> Web Front End for SNMP Queries</H1>
<FORM METHOD="GET" ACTION="snmpquery">
<P> What host?
  <INPUT TYPE="TEXT" NAME="hostname" SIZE=20>
</P>
<P> What variable?
  <INPUT TYPE="TEXT" NAME="variable" SIZE=40>
</P>
<P> What instance?
  <INPUT TYPE="TEXT" NAME="instance" SIZE=4 VALUE="0">
</P>
<P>
  <INPUT TYPE="SUBMIT" VALUE="Run Query">
</P>
</FORM>
</BODY>
</HTML>
```


And its accompanying processor:

```
#!/usr/bin/perl5
use SNMP;
use CGI qw(:all);

print header;
print start_html(-title=>"Results of SNMP query");
print h1("Results of SNMP query");

$hostname = param('hostname');
$variable = param('variable');
$instance = param('instance');

if ($sess = new SNMP::Session(DestHost=>$hostname)){
    $value = $sess->get([$variable, $instance]);
    $error = $sess->{ErrorStr};
    if ($error){
        print p(b("Error: $error\n"));
    }
    print p("Value of $variable.$instance for $hostname is $value\n");
}
else {
    print p("Error: could not bind $hostname\n");
}

print end_html;
```

No points will be awarded for finding any of the obvious shortcomings of this little system – except the sudden appearance of something called an “instance.” At this point, we’re still dealing with things that are one to a device; each device has exactly one system description, for example. That means that so far I’ve been able to get away with throwing around `system.sysDescr` and `system.sysDescr.0` without ever quite mentioning where the trailing `.0` comes from. However, SNMP also deals with things that come in multiples. For example, later on we’ll probably be interested in `interfaces.ifTable.ifEntry.ifDescr`, which describes a network interface. Most network devices of interest have multiple network interfaces, and so there are multiple instances of `interfaces.ifTable.ifEntry.ifDescr`, one for each interface. I could have just left this hard-coded at 0 for now, but then I would have had to reprint a slightly modified program when we got to multiple-instance variables, and that would be annoying.

Next time: We move on to more interesting parts of MIB II, and branch out into more interesting programming.

NT Accounts: A Starting Point

I have often been asked about the different types of accounts in Windows NT. This time I'll talk a little bit about each of them, how they differ, and what they can be used for.

NT and UNIX Parallels

I find it's simplest to describe Windows NT accounts to experienced UNIX users by using NIS as a reference. To begin, think of the NT domain as an NIS domain (and for those of you who care, NT5 domains are similar to NIS+ domains). Similarly, the NT Security Accounts Manager Database (SAM) is roughly equivalent to the UNIX passwd file. An NT Member server is roughly equivalent to a UNIX server which is NOT an NIS master or slave (but is part of the NIS domain). A Windows NT Primary Domain Controller (PDC) is roughly equivalent to a UNIX NIS master, and a Windows NT Backup Domain Controller (BDC) to a UNIX NIS slave server (NOTE: Unlike UNIX slave servers, the BDC does *not* have its own separate SAM; it uses the Domain SAM exclusively). Whew! That's a lot of relationships! See the table on the right for reference.

Trust

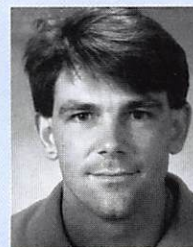
Before we go any further, we have to talk about "trust" and how it is used in Windows NT Domain structures. When one NT domain "trusts" another, then the accounts and groups defined in the "trusted" domain can be used for authentication and authorization in the "trusting" domain. For example, let's say the account "pcc" is a member of the Windows NT domain "NTS" (noted by the syntax "NTS\pcc"). Further, let the Windows NT domain "IWI" trust "NTS" (so that NTS is *trusted*, and IWI is *trusting*). Then I can log on to a machine that is part of the IWI domain with the account "NTS\pcc". The IWI domain would know to contact the *trusted* domain and validate my logon credentials. For the sake of this article, it is important just to understand that Windows NT domain accounts can be used in other domains, if a trust relationship has been established.

Account Types

With these analogies, let's look at the three types of Windows NT accounts and their usage.

The first type of account is the Local account. It is like a user account in the `/etc/passwd` file on the UNIX workstation. Although the machine is part of an NIS domain, the accounts in the local `/etc/passwd` file are valid for that machine alone. This is true for NT as well. Any account defined in the SAM of the individual workstation or member server is for that host alone. Local accounts are *only* valid when used on the machine for which they were created. There is also a special class of local accounts that are called "built-in." These accounts are used mostly for system purposes and are not usually modified. Some of the built-in, or special accounts, are: Creator/Owner, System, and others. These accounts are seen when setting access control entries, but are *not* seen when managing regular local user accounts.

The second type of account is the Domain account. These accounts are defined in the SAM on the PDC, much as NIS accounts are defined in the `/etc/passwd` file on the NIS master server. These accounts, like NIS accounts, are valid for any/all machines that participate in the domain, and any domain for which this domain is *trusted*.



by Phil Cox

Phil is a member of the Computer Incident Advisory Capability (CIAC) for the Department of Energy. He also consults and writes on issues bridging the gap between UNIX and Windows NT.

<pcc@ntsinc.com>

<u>Windows NT</u>	<u>UNIX Equivalent</u>
NT Domain	NIS Domain
SAM	password file
PDC	NIS Master
BDC	NIS Slave
Member Server	Server that uses the NIS domain

There are three type of Windows NT accounts:

- *Local (in SAM on individual Workstation or Member server)*
- *Domain (in SAM on PDC and replicated to the BDCs)*
- *Domain Local (in SAM on PDC, but are restricted to the PDC and BDCs).*

Just like in UNIX, they differ in their scope and in their administration.

The third type of account is the Domain Local account. It is actually a Local account on the PDC as defined above. Since the machine is the PDC, and the PDC's SAM is the SAM for the BDCs as well, this account becomes "local" to the PDC and all BDCs in the domain. Normally, a user defined in the SAM on the PDC would become a "Domain User" and would have the characteristics (described in the paragraph above) of such an account. A Domain Local account, however, is specially marked and is not seen in the regular list of "Domain Users."

Which Account To Use

You might wonder, "When might I choose one account type over another?" Local accounts are typically used when you desire to restrict access to an individual machine. Domain accounts (by far the most common) are applicable when you want to allow access to a wide range of resources. Finally, use Domain Local accounts when you require an account that will have access to resources on Domain Controllers but isn't trusted outside of the Domain itself.

All NT accounts are administered through a "User Manager," but, depending on the account, you'll need the right manager! All local accounts are administered through the "User Manager" (`usrmgr.exe`) directly on the host itself. All domain-related accounts are administered with the "User Manager for Domains" (`usrmgr.exe`) from any host that has the program and has permission to connect to the domain.

While the naming convention may be new, many of the same concepts that apply to accounts in UNIX also apply to NT. If you've mastered the fundamentals of UNIX logins and NIS domains and servers, you're well on your way to understanding NT accounts and domains controllers.

Knit One, Perl 5.005

There's a New Perl on the Block

The latest “major” revision of Perl, version 5.005, is now available. I'll discuss some of its new features, and some of the less stable experimental additions to the language.

Keep Your Feet Dry

Although the release of a major revision is exciting, you shouldn't immediately download Perl 5.005 and install it on your system. In a production environment where users depend on the reliable operation of a large set of Perl modules and programs, this would be a disaster. For one thing, Perl 5.005 is *binary incompatible* with previous versions of Perl. This means that any C- or C++-based modules that are installed in your Perl tree must be rebuilt for Perl 5.005. Some of those modules may not even compile against Perl 5.005 because the Perl API has changed; therefore, you may have to wait for new 5.005-compatible versions of those modules to be released. Another issue is that while maintainers have generally taken great pains to ensure backward compatibility with existing Perl scripts, incompatibilities due to changes and/or bugs no doubt *do* exist.

If you are running Perl on a machine on which you are the only user, or where it is not being used for critical applications, you can convert to Perl 5.005 by taking an inventory of what modules are installed on your system (perhaps with the help of `perldoc perllocal` or the CPAN module's `autobundle` command), building Perl 5.005, then reinstalling all of the modules that you want to keep.

On the other hand, in a production environment where a stable Perl is a critical part of daily operations, you will have to create an independent Perl 5.005 installation, reinstall the necessary modules there, then have your users test their applications against this separate installation. Applications that won't run under 5.005 can be fixed, or if that isn't practical, “hardwired” to use an earlier version of Perl kept around for backward compatibility. Only after everything is tested and stable – a process that might take weeks or months – should you make the newer version of Perl your new standard.

It's just a suggestion, but if I were contemplating moving a production environment to a newer version of Perl, I would wait a few months and hope for more maintenance releases of Perl 5.005 in the interim.

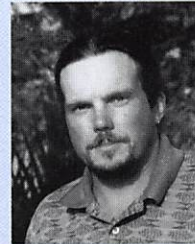
What about Threads?

Perl 5.005 was originally supposed to include threads support. To a certain extent, it does, but as of 5.005_02, threads are still classified as an experimental feature. I'd like to discuss threads programming in Perl, but the time is not yet right.

Okay, What about the Compiler?

Development on the Perl compiler (also known as B) continues. While the compiler itself is still considered an experimental feature, there are a number of compiler-related modules that are already useful to developers. It's also instructive to get a quick overview of the compiler.

The Perl compiler works by parsing Perl source code into an intermediate representation. Then, with a suitable “backend,” it transforms the intermediate representation into something else. The CC backend, for example, creates optimized C code. The resulting C



by Joseph N. Hall

Joseph N. Hall is the author of *Effective Perl Programming* (Addison-Wesley, 1998). He teaches Perl classes, consults, and plays a lot of golf in his spare time.

<joseph@5sigma.com>

This column refers to the most recent version available as of this writing, Perl 5.005_02. There may have been additional releases by the time you read this article, but you shouldn't expect any radical advances or changes in the immediate future.

There have been quite a few internal changes to Perl regular expressions in Perl 5.005. Regular expressions are now more efficient, a number of bugs have been fixed, and the regular expression code in Perl itself has been somewhat cleaned up.

code must then be compiled with a C compiler to produce an executable program. For example, here's a short program, which we'll call `cc-test.pl`:

```
print "Welcome to compiled Perl!\n";
for $i (1..10) {
    print "$i times $i is ", $i * $i, "\n";
}
```

You can produce a corresponding C program, `cc-test.c`, with the somewhat unintuitive command:

```
% perl -MO=CC,-occ-test.c cc-test.pl
```

From there, you need to compile and link this C program. You can do it directly using the `cc_harness` program, but the process is simpler if you use another program called `perlcc`. `perlcc` takes care of the intervening steps automatically. For example:

```
% perlcc cc-test.pl
```

will first run the Perl compiler to produce C source code, then compile and link it into an executable called `cc-test`. When you run `cc-test`, it works just as if you were running the original Perl source in `cc-test.pl`.

This is nice as far as it works, but as the documentation states repeatedly, the Perl compiler is buggy and still experimental. However, there are some compiler-related features that you may find interesting. For example, there is another backend called `Xref` that can be used to produce a (somewhat cryptic) cross-reference listing of some of the contents of the Perl source:

```
% perl -MO=Xref cc-test.pl
cc-test.pl syntax OK
File cc-test.pl
Subroutine (definitions)
  Package UNIVERSAL
    &VERSION s0
    &can s0
    &isa s0
Subroutine (main)
  Package main
    $i 5, 5, 5, 5
    *i 4
```

You might also find the `Lint` and `Deparse` backends useful. For more information about the compiler, use the `perldoc` command to read the documentation for `B`, `B::CC`, `O`, `perlcc`, and follow the references to other modules and programs as necessary.

Not So Regular Expressions

There have been quite a few internal changes to Perl regular expressions in Perl 5.005. Regular expressions are now more efficient, a number of bugs have been fixed, and the regular expression code in Perl itself has been somewhat cleaned up. There are a fair number of user-visible changes as well. First, there are some new regular expression atoms. For example, there are now atoms for pattern lookbehind:

```
@foos = /\bfoo.*?\b/g;      # find all words beginning with foo
@nofoos = /(?!<=bfoo).*?\b/g; # find all words beginning with
                                # foo, but leave off the foo
```

There are also atoms for “conditional” expressions, atoms that affect backtracking, and even atoms for executing arbitrary Perl code as a regular expression is matched.

Another addition is the `re` pragma module, which controls various aspects of regular expression behavior. One use for `re` is debugging output:


```
% perl -Mre=debug -e '/".*?"/'
compiling RE '".*?"/'
size 8 first at 1
1: EXACT <">(3)
3: MINMOD(4)
4: STAR(6)
5: ANY(0)
6: EXACT <">(8)
8: END(0)
```

The `qr//` (regular expression quoting) feature greatly simplifies the efficient use of regular expressions that aren't known until run time. The contents of `qr//` are a regular expression "string" (the same as in an ordinary match operator `m//`). The result of `qr//` is a scalar that can be used standalone like a match operator, or incorporated efficiently into other regular expressions:

```
print "Field to match: ";      # For example, 'subject'
chomp($field = <>);
$re = qr/$field/i;           # /i means ignore case
print "Type in a line: ";    # For example, 'Subject: some perl
                             # stuff'
print <> =~ /$re: (.*)/, "\n"; # prints 'some perl stuff'
```

The new features are documented in the `perlre` man page, the `re` module's documentation, and the `qr//` entry in the `perlop` man page.

Lots More New Stuff

There are many more small changes and improvements in Perl 5.005. Here are some of the more stable (nonexperimental) ones.

- `foreach (1..1000000)` is now optimized so that it does not construct a temporary list one million elements long.
- You can now use `foreach` as a statement modifier. For example:


```
print "The number is $_\n" foreach (1..10); # prints 10 numbers
```
- The `fields` and `base` pragmas allow you to add more compile-time semantics and checking to your Perl classes.
- Perl is more tolerant of carriage returns in source files. DOS-formatted (CR-LF) Perl source used to cause fatal errors; this is no longer the case.
- Tied arrays and handles are better supported.

For More Information

For more about what's new in Perl 5.005, check the documentation. You can build Perl 5.005 and read the `perldelta` man page, or you can inspect it online at any CPAN mirror site, for example, [<http://www.perl.com/CPAN-local/doc/manual/html/pod/perldelta.html>](http://www.perl.com/CPAN-local/doc/manual/html/pod/perldelta.html).



by Daniel E. Singer

Dan has been doing a mix of programming and system administration since 1983. He is currently a system administrator in the Duke University Department of Computer Science in Durham, North Carolina, USA.

<des@cs.duke.edu>

Toolman: Generating Web Pages with `sh` and `make`, Part 1

Yeah, yeah, I've been hearing plenty lately about using Perl to generate Web pages. But I've been doing the same thing for a while now with the humble `ol'` shell. Basically, I use the Bourne shell and its intrinsic string-based processing as a macro-processing language to generate HTML. Arguably, Perl or even `m4` or various other scripting or macro languages [1] would be better suited for this purpose, but unless your Web pages are going to be fairly complex or extensive, good `ol'` `sh` does quite a nice job, thank you. Throw in some appropriate scripting and a `makefile`, and you can automate processing – and even generate various versions of the same pages, for example, for prototyping or for different network sites.

Dubious as these claims may seem, the techniques I'm going to describe are really quite practical. What it gets down to, though, is that I'm just one of those incorrigible (and unrepentant) `sh` hackers at heart. So if you've got a hankering to do some shell and `make` programming to whip out consistent, easily maintained HTML code, please read on. And if you're one who prefers an alternative scripting language, these concepts are likewise applicable.

Due to space considerations, we'll cover this material in two parts; fortunately the content easily accommodates this. In this first installment, we'll talk about the shell, and in the February issue of *login:*, we'll cover `make`.

Tutorial Or Not Tutorial?

This article will *not* be a tutorial on shell programming or on writing makefiles (or on writing Web pages, for that matter), but will demonstrate how these tools can be applied in the context of Web page generation. And by generation of Web pages and HTML, I mean the pregeneration of static Web pages only, not dynamic generation or CGI techniques. (The latter are within the realm of possibility, I just haven't messed with it.) Please note that my use of these methods is a work in progress, so you will likely be able to improve upon what is presented here.

Objectives: Consistency, Simplification, Automation, Mutability

We want to satisfy several objectives with our choice of tools for Web page generation. One is to provide consistency to the appearance of, and the data in, the pages. This can be accomplished by the use of variables to hold definitions, such as colors and addresses, and by the use of user-defined functions to generate constant or similar sections of HTML, such as page headers and footers. Another objective is simplification. Again, shell functions can be used to consolidate much of the tedium of writing, say, an HTML table. A third objective is automation – having to go through as few steps as possible to get from one state to the next, in particular, from our source files to our final HTML documents. A fourth objective is mutability: we might want to create different versions of the same Web pages, for instance, for two separate servers at separate sites. `make` becomes useful for these latter two objectives.

Why Shell?

I never really thought of the shell as a macro processor until I started using it in this context. But when you get down to it, the shell is very heavily built around string processing [2], which is what macro processors are all about, right? A line in a script is interpreted as a command only after variables and various other substitutions are interpolated. A shell variable behaves much like a simple macro. And a shell user-defined function can be tantamount to a macro with parameters. Plus you get "all that good stuff," the usual benefits of the shell: control structures, pipes, easy access to external commands, environment variables, etc. Personally, I find HTML somewhat tedious to write. Even comments are awkward, and are much easier in shell syntax. For me, the shell provides a much more comfortable (familiar?) style.

Here's a simple example of how you might use shell code to generate some HTML. A variable can be defined to hold a value or some HTML code:

```
HAPPY="<EM>Let's get happy!</EM>"
```

and then that value can be written out at any following point in the shell code with:

```
echo "$HAPPY"
```

Unless you're planning on saying that a lot of times, this construct isn't going to be terribly productive. A more useful approach is to use a function, such as:

```
emphasize() {
  echo "<EM>$*</EM>"
}
```

This function can then be invoked with:

```
emphasize "Let's get happy!"
```

The `$*` in the function body is replaced by the parameter(s) to the function, and the function is reusable with other text. And here's an even more general form:

```
tag_data() {
  _TAG="$1"
  shift
  echo "<$_TAG>$*</$_TAG>"
}

emphasize() {
  tag_data EM "$@"
}
```

And if we *were* going to be saying it alot, then adding this might even make sense:

```
get_happy() {
  emphasize "Let's get happy!"
}
...
get_happy
```

I hope you're starting to get the idea.

How the Processor Works

One approach to using the shell as an HTML-generating macro processor is to write a master script consisting of your standard definitions (variables) and macros (functions), and then have the master script *source* (the shell's period `.` built-in command, see `man sh`) the files that define individual Web pages, producing a browser-ready HTML output file for each input file.

So, to elucidate, each Web page is derived from a source file of shell code consisting largely of calls to the functions defined in the master script, and possibly redefinition of

...a shell user-defined function can be tantamount to a macro with parameters.

some of the variables and/or functions. Unique features for each Web page are specified through parameters to functions, variable redefinitions, choice and order of function calls, and/or specific HTML coding. Optionally, local functions and other special processing can also be added to the source files.

Here's some code from a script named `gen_html.sh` that I used recently to help generate a group of HTML files for a talk. The script begins with about 700 lines of variable and function definitions, option processing, etc. (Many of the variables can be overridden by variables in our HTML source files or by environment variables; the potential here for wrapper scripts is strong.) Finally, the loop below occurs at the end of the script. In it, we just process any arguments left on the command line, which should all be the names or basenames (suffix omitted) of HTML source files. (Actually, by this point, the arguments have already been scanned once to look for errors, and, if there were no arguments, a "-" indicating standard input would have been pushed onto the positional parameters.) Each input file is processed, and a corresponding HTML output file is produced:

```
##
## process (source) each input file
##
for FILE do
  case "$FILE" in
    # for stdin, make a tmp file and just process to stdout
    -)
      : ${UPDATE_DATE=`date '+%e %B %Y'`}
      HTML_FILE="???.$OUT_SUFFIX"
      [ "$QUIET" = 0 ] && echo "$PROG: processing stdin..." >&2
      cat > "$TMP_FILE"
      . "$TMP_FILE"                # source it
      rm -f "$TMP_FILE"
      ;;
    # otherwise, process and output to filename.html
    *)
      case "$FILE" in
        *."$IN_SUFFIX")
          OUT_FILE=`echo "$FILE" |
            sed 's/.'"$IN_SUFFIX"'$/'.'"$OUT_SUFFIX"'/'`
          ;;
        *)
          OUT_FILE="$FILE.$OUT_SUFFIX"
          if [ ! -f "$FILE" -a -f "$FILE.$IN_SUFFIX" ]; then
            FILE="$FILE.$IN_SUFFIX"
          fi
          esac
          HTML_FILE="$OUT_FILE"
          [ "$QUIET" = 0 ] && echo "$PROG: processing $FILE" >&2
          {
            UPDATE_DATE=`get_update_date "$FILE"`
            case "$FILE" in
              /*)
                . "$FILE"                # source it
                ;;
              *)
                . ./"$FILE"              # source it
                esac
            } > "$OUT_FILE"
          esac
        done
      exit 0
```


This loop could perhaps be simpler, but, as written, it can work as a filter processing from standard input to standard output, and it can handle filenames with or without a source file suffix. Some date processing also takes place so that a "Last updated:" time-stamp based on file modification time can be added to each page. (You know how hard it otherwise is to remember to manually update those dates!) The total length of this script might sound excessive, but it can be reused with minimal modification, and so the initial investment can pay off repeatedly.

This script might be invoked with a command line like:

```
./gen_html.sh index part1 part2 notes
```

Here's a very trivial example of what an HTML source file might look like:

```
## @(#) test.hsrc
## 9/98, D.Singer

begin_doc
begin_head -t "This is a test of `gen_html.sh`"
end_head
begin_body -bg "skin.jpg"
do_break
heading -C 1 "This Is A Large Title"
heading -C 4 "and this a more subtle title"
do_break
do_hrulerule -s 6 2
do_break 2
begin_center
begin_font -s +3 -c red
emphasize "Thanks for coming!"
end_font
end_center
do_break 2
do_hrulerule -s 6 2
do_break
do_last_update
do_break
do_footer
end_body
end_doc

## end of hsrc file
```

And here's the resultant HTML output (slightly altered to conserve space):

```
<HTML>
<!-- html document <www.cs.duke.edu/~des/workdir/test.html> -->
<!-- generated on Mon Sep 21 00:09:31 EDT 1998 -->
<!-- via `gen_html.sh' -->
<HEAD>
<BASE HREF="http://www.cs.duke.edu/~des/workdir/">
<TITLE>This is a test of `gen_html.sh'</TITLE>
</HEAD>
<BODY>
BGCOLOR="#ffffff"
BACKGROUND="skin.jpg"
TEXT="#000000"
LINK="#339999"
ALINK="#BBBB11"
VLINK="#336060">
<BR>
<H1 ALIGN=CENTER>This Is A Large Title</H1>
<H4 ALIGN=CENTER>and this a more subtle title</H4>
<BR>
```

Got a tool that's useful,
unique, way cool? Please send a
description to <Toolman@usenix.org>.

Using the shell as your HTML generator provides all of the features and power of the shell, and of course the whole toolbox of UNIX utilities . . . that comes along with it.

```
<HR SIZE=6><HR SIZE=6>
<BR><BR>
<CENTER>
<FONT SIZE=+3 COLOR=red>
<EM>Thanks for coming!</EM>
</FONT>
</CENTER>
<BR><BR>
<HR SIZE=6><HR SIZE=6>
<BR>
<FONT SIZE=-1>
Page last updated: 21 September 1998
</FONT>
<BR>
<FONT SIZE=-1>
URL: http://www.cs.duke.edu/~des/workdir/test.html
</FONT>
<BR>
<FONT SIZE=-1>
Copyright &copy; 1998, Daniel E. Singer. All rights reserved.
</FONT>
</BODY>
</HTML>
<!-- end of generated document -->
```

As you can see, the master script can also generate HTML comments for each Web page to provide standard identification blocks or other metadata. A more complete example would include tables and other snazzy features. Unfortunately, we don't have the luxury here of the space that would be required. But I can't resist at least showing you what some HTML source might look like for a table:

```
begin_table -cols 2 -w 80% -C
table_data -fs +2 -R "<EM>Row 1"
table_data -nobr "This table has two columns, 80% width, and is
  centered. The first column is right justified and has a bigger
  font. For the second column, we're forgoing putting a break
  between each line."
table_data -fs +2 -R "<EM>Row 2"
end_table_row
table_data -fs +2 -R "<EM>Row 3"
table_data -nobr "The second row only had 1 column."
end_table
```

As the code shows, this table is self-documenting! It's still a long way from WYSIWYG, but I like it better than the HTML that it will generate. See the end of this article for a URL for a sample `gen_html.sh` script; it will include the HTML-table-generating shell code.

Other Benefits of the Shell Approach

Using the shell as your HTML generator provides all of the features and power of the shell, and of course the whole toolbox of UNIX utilities (`grep`, `sed`, `awk`, `ls`, `date`, etc.) that comes along with it. An example of this is a situation where I use a flat database text file (lines with tab-separated fields) along with a supplementary script to produce multiple Web pages, each containing a list derived from the database, each sorted on a different key, and each providing hypertext links based on one of the fields. Believe me, this is much easier than maintaining these separate HTML pages by hand. For the situation I have in mind, a list of alumni email addresses gets sorted by name and by class (that is, year). And a similar database exists for Web addresses. Here's how it works.

First, there's the database (a text file) that looks like this (the names have been changed to protect the innocent):


```
# email-addr.db
# @(#) /u/des/public_html/sf/email-addr.db 1.29
Abbott, Gail      gabb@mail.ced.net 1991
Adams, Albert    adman@fisheads.com 1976
Adams, Fred      adams@nunez.org 1982
...
```

A supplementary script named `cvt_addr.sh` reads the database, and then, depending on selected options, writes out the records in HTML form, sorted, with email addresses converted to `mailto:` links, and with section `NAME` anchors added. In the appropriate spot in the HTML source file, the line:

```
./cvt_addr.sh -f email-addr.db
```

invokes the script, and inserts the HTML data derived from the database sorted by last name. In another HTML source file, the same script is called with the addition of the `-c` option to get a sort by class. Additional code in each of these HTML source files generates the jump lists used to go to a `NAME` anchor for a particular year or letter of the alphabet. For instance:

```
echo "<A"
NEXTA="<-<A"
for LETTER in A B C D E F G H I J K L M N O P Q R S T U V W X Y Z;
do
[ "$LETTER" = "Z" ] && NEXTA=
echo "HREF=\"email-addr.html#$LETTER\">$LETTER</A>$NEXTA"
done
```

This is much more concise and maintainable than the equivalent written-out HTML.

Other Shells

Of course, this could all be done with shells other than Bourne shell, such as `ksh`, `bash`, or even `zsh`. In fact some of these would probably make the job easier, as Bourne tends to be a bit archaic in some ways, and they are worth investigating. (I'm just too lazy.) I do tend to avoid the `csh` derivatives for scripting, but if you don't want to take my advice on this one, well, let's just say you're on your own . . .

Until Next Time

That's it for the shell half of our discussion. To get the whole scoop, you'll just have to bite your fingernails in anticipation until the February issue of *login*, when we'll explore how `make` plays an integral role in this process, providing the significant capabilities of automation and mutability. Please tune in next time.

URLs:

```
<http://www.cs.duke.edu/~des/toolman.html>
<ftp://ftp.cs.duke.edu/pub/des/scripts/gen_html/INDEX.html>
```

Notes

[1] I've even seen a recent article about using `cpp` (the C pre-processor) and `make` to do something very similar (Jim Fox, "Unity Among Web Pages" in *SunExpert Magazine*, August 1998, pp. 42-45.), though there are some substantial differences in style and content between these two approaches. But, shoot, for us shell diehards, well, need I say more?

[2] This is despite Bourne shell's notable lack of built-in string processing functions. For many operations, it is necessary to engage external commands such as `awk`, `sed`, `expr`, etc.

Of course, this could all be done with shells other than Bourne shell . . .

source code UNIX

Help a Friend Get Online, Cheap



By Bob Gray

Bob Gray is co-founder of Boulder Labs, a digital video company. Designing architectures for performance has been his focus since he built an image processing system on UNIX in the late 1970s. He has a PhD in computer science from the University of Colorado.

<bob@boulderlabs.com>

You probably have a friend or relative who has expressed a desire to use the Web or communicate using email. They may want to shop for books online, monitor their financials, or search for health information. They might figure that the entry cost for equipment would be \$1500 – \$2500 – for many, a prohibitive amount. With some time, initiative, and willingness to take the path less traveled, however, a system capable of Web, email, and even document processing is possible for a small fraction of this price.

The useful life of a computer in the business world is about three years. After that, current versions of mainstream office software run sluggishly on old hardware. The industry is geared toward getting companies to replace perfectly good hardware for the latest, fastest, biggest equipment. (Reminds me of the '60s and '70s, when many would replace their new cars every few years.)

So here's the plan: We're going to find some older equipment and install Source Code UNIX software on it. Then we'll configure it in a way to give your friend or relative the capabilities he or she needs. We'll discuss how to connect to the Internet at an appropriate price/capability level.

Disclaimer: The ideas presented here are not meant to balance your personal time/money equation. Think of this proposal as an "exercise for the student" or as time invested in a hobby. (Otherwise, you might instead spend your energy on payable consulting and with the proceeds just buy an iMac outright for your friend.)

Is This Heresy?

For your friend who has more surplus time than cash, there are alternatives to buying that new, flashy, well-advertised, \$2000 Wintel system. One reasonable option for some is the new \$1300 Apple iMac. And the sub \$1000 Wintel systems will probably be available by Christmas. Source Code UNIX running on one-generation-old equipment is a good solution for many. You and I know that the Pentium-133 of 1995 runs UNIX just fine. For those who primarily surf the Web and use email, more power, large memories, and huge disks are just a waste. We may be concerned about putting the combination in the hands of a novice, but I surmise that keeping such a system running for a novice is about the same amount of work as keeping a new Windows 98 system running for the same person. Further, your friend won't be faced with the constant upgrade costs for the OS, word processing, spreadsheets, and other applications. Once a UNIX system is set up, it should run trouble-free. It can be left on, ready to use any time like most household appliances. Your friend won't be faced with the problems typical of Wintel systems – constant crashes and the periodic, time-consuming "reloading" the system.

Yes, there is potentially tons to learn about a UNIX system but *you* are going to spoon feed the required pieces to the novice. People's learning and coping skills are amazing – they can handle UNIX with about the same effort that doing the same tasks on other systems requires.

Many applications in the Ports Collection probably will be useful to your friend. The October *login*: article, "Application Software: Ports and Packages" <www.boulderlabs.com> describes many of these programs. Some of the likely candidates include the general image manipulation software, GIMP (similar to Photoshop), drawing programs like TGIF; plotting packages such as GNUPLOT; spreadsheet software such as *sc* or *oleo*; editors, including the WYSIWYG *Textedit*; and a large collection of games.

A small amount of experimenting with the window manager configuration should yield something that our novice will be happy with. (If necessary, you can configure `fvwm` to mimic Win 95.) What about word processing? If they need word processing (and a printer), things become more complicated (as it does for commercial software.) Fortunately, several office packages run just fine on Linux or the BSD variants. StarOffice, Applix and Corel are the primary suppliers for this add-on, commercial software. They are reasonably priced; some are free from the Internet.

Getting the Equipment

First, let's outline the assumptions. We're trying to get a basic amount of functionality without spending lots of your friend's money. They are going to use this system for personal use; none of it is business-critical. Make sure they could live with the inconvenience of losing their system as a result of hardware failure. (You'll teach them how to save their precious things on floppies.) It's not a software-development machine or graphics workstation; these are beyond the scope of this article. Your friend will be patient while various things get ironed out. *You* have some spare time to build and to configure the system and, then, some time to help and advise.

I suggest finding a low-end Pentium machine. Sure, there can be exceptions; you might happen to have some spare Alphas around that could run Linux, or you might have a working SparcStation-2 running SunOs4.x that is ready to be donated. But for most, let's find that Pentium. Some of the 486s, given enough memory, will work fine, but if you don't already have one to use, I wouldn't bother with something that old. A Pentium-90 or Pentium-133 with 24-32 MB is a good choice. Here is where the initiative comes in; depending on the available budget, you or your friend might want to visit companies to obtain this hardware. A nice, friendly attitude at the right place and time could result in a gift. (After all, what's the company going to do with machines that can no longer run Microsoft Office 98?) Maybe \$50 would make their accountants happy. Larger companies sometimes sanction obsolete-equipment sales for employees and for the public. Here is an opportunity to negotiate price.

You could look in the classified section, but often people think that because a system cost them \$3,000 three years ago, it is worth half that today – False! Think 5%, certainly no more than 10%. Used computer stores have appropriate equipment. They tend to be expensive, but it's worth a try. Department stores, OfficeMax, etc., periodically need to clear out unused obsolete models and display equipment. Hunt around. Deals abound for leftovers. I'm looking at a flier now (early October) that advertises a Cyrix 133/32MB/2.1GB/33.6Fax/32xCD/14"SVGA for \$539. If the graphics card is acceptable, this is everything needed. For a little more, there is a K6 233MHz MMX, S3 ViRGE 4MB video, 32MB/2.5GB/32xCD/33.6 system for \$699. It still needs a monitor (anywhere from an extra \$100 to \$1000), but it's a very hot machine for Source Code UNIX – enough power even for most developers.

The Internet offers resources for new equipment. I like <<http://www.insight.com>>. These guys pick up retailers' and manufacturers' surplus equipment for a song and try to unload it quickly for a profit. They have a number of Pentium II (233, 266, 333) systems for under \$1000. You'll need to add a monitor. These systems don't really qualify as "obsolete" hardware, but they might suit some people who have some money. Push their "specials" tab, then go to "inventory blowout." Fill in the form with category "Computer Systems" and a maximum price. This will pop up a list of systems such as IBM Aptiva, Compaq Presario, HP Vectra, and others. Look at the technical specifications and the stocking status. You'll see choices in the \$500 - \$800 range. They also have an auction system for the adventuresome. Note the risks of buying off-brand or non-warranted equipment; if something breaks, you're on your own.

*Many applications in the Ports Collection probably will be useful to your friend. See the October ;login: article, "Application Software: Ports and Packages" <www.boulder-labs.com>. Some of the likely candidates include the general image manipulation software, GIMP (similar to Photoshop), drawing programs like TGIF; plotting packages such as GNU PLOT; spreadsheet software such as *sc* or *oleo*; editors, including the WYSIWYG Textedit; and a large collection of games.*

The list of useful free or inexpensive software that can run on these systems is endless. There are programs to balance checkbooks, track stocks, run image scanners, and manipulate audio. The games doom, chess, backgammon and others are all available. Once your friend handles the basics of surfing and emailing, you can enhance their software suite.

Check out the “computers” section of <http://www.surplusdirect.com> for new and refurbished equipment in all price ranges. Their new \$549 Worldnet 7416 with a 233MHz K6 is a good candidate. These guys also have auctions.

Other good places to shop include <http://www.streetprices.com> and <http://www.pricewatch.com>. My June *login*: article, “PC Hardware for Source Code UNIX” <http://www.boulderlabs.com>, lists many other Web sites.

Most Pentium, AMD or Cyrix hardware will work fine with Source Code UNIX. As I mentioned in the June article, the problem areas are mostly cheap, off-brand video cards and some Plug-and-Play devices. Some early research or try-before-buy attitude will save later headaches.

Loading and Configuring the System for Basic Capabilities

Get your software distribution CDs ready. The favorite supplier for Linux is RedHat 5.1 for \$50. For those preferring a BSD UNIX, consider FreeBSD 2.2.7 (or 2.2.8 by the time this is published) for \$40. The main Web locations for Source Code UNIX are (you’ll need software from only one of these): <http://www.linux.org>, <http://www.infomagic.com>, <http://www.redhat.com>, <http://www.caldera.com>, <http://www.freebsd.org>, <http://www.netbsd.org>, and <http://www.openbsd.org>.

My article in August *login*: “Loading Source Code UNIX,” goes over the general operations that are needed to bring up the system. This section augments it with material oriented towards getting certain capabilities for the novice. The first issue to handle is: What “look and feel” do we want for our friend? If the person has been exposed to Windows, it may be best to try to mimic that GUI. You can get the “Explorer” look and feel with programs such as `xm`. Hackers have munged the `fvwm2` window manager to look like Windows 95. Lots of the idioms, such as window frames and task bars, are present. There are file managers that allow drag-and-drop operations. I don’t think it’s very important to tightly copy the Windows 95 environment, but this issue is open for discussion and experimentation.

For the novice who hasn’t been “preconditioned,” there are choices. `Fvwm` and `ctwm` are both good window managers that can handle everything a novice needs. You, the instructor, have the opportunity to teach whatever you believe in. For example, I prefer windows that activate when the cursor is placed in them versus the click-to-type kind. I hate “autoraise” windows, and I program F1 and F2 to raise or lower the window that the cursor is in. I would avoid features that can get a person lost, such as a virtual window manager that allows you to scroll to empty desktops. The aesthetic decisions go on and on – you get the idea. This is the easy stuff. How are you going to teach file manipulation? Drag and drop for everything or possibly some command line operations? How will the user back up precious files to floppy? Sorry, I don’t have all the answers – it depends a lot on the user.

How will they browse the Net? The current versions of Netscape are memory pigs, but they probably represent the best option. I’d stick with the slightly smaller Netscape Navigator that doesn’t have as much bloatware as Netscape Communicator, unless you want the mail system and a news reader all in one. The standard versions come with 40-bit encryption. Some financial institutions (<http://www.vanguard.com> for example) require the much stronger 128-bit encryption to access customers’ data. Find out what your friend intends to browse to learn if the US-limited, strong encryption version is necessary.

What mail system makes sense for a novice? I know of many beginners who do just fine with Pine. I swear by `exmh` and don’t think it would be too hard for a smart, noncomputer person. Lots of people like using the mail facility in Netscape.

What editor will you set up? I think there are many choices better than `vi`. Consider `Textedit`, `Jot` and other modeless editors. The office packages discussed in the next section offer other choices.

You probably should set up your friend as a normal user; I think you would regret starting them as root. For operations for which they need more privileges, give them the `sudo` command or make some `sudo` programs. But don't flood them with too much superfluous information too soon.

You probably want their login scripts to start a window manager that has icons for mail and Netscape. Other processes such as `xclock` and `calendar` might also be useful.

You should "know your audience." You decide how much or how little they can handle. There are individuals that will never feel comfortable with computers and software – even Macintoshes. Maybe you won't want to get involved with such cases. Others, like my 12-year-old niece, can handle just about anything. When the family Wintel machine crashed and burned, she had no hesitation wiping the disk and reloading. If this non-geek girl can get that far with Wintel, she won't have any problem with a "real" operating system.

Connecting to the Internet

Internet Service Providers (ISPs) are everywhere. Some advertise heavily, others are discovered by word of mouth. Most ISPs offer part-time dial-up connectivity (10-15 hours) for about \$10 per month and a bigger plan or unlimited usage for \$20 or so a month. In general, I would suggest using Point to Point Protocol (PPP). Just about every ISP can deal with PPP, and the protocol can handle dynamically assigned IP addresses. How does one choose a service provider? I would evaluate the following factors as a function of the price:

- How much technical support is available?
- Will they work with you even though you don't have Windows 95?
- Is it a local call?
- How much time do you get per month?
- What is the incremental charge when you exceed your quota?
- Is the service reliable, or does it often go up and down?
- How available are the dial-in modems when you would be using the service?
- Is the ISP machine reasonably loaded in terms of the number of users and the bandwidth available to the Internet?
- Can you get a reasonable email address like `<robert@clark.net>` or must you live with something obscure like `<rbh4782@aol.com>`?
- What dialin speeds are offered?

PPP Setup

Modems, IP addresses, routing, and name services might be the most difficult and frustrating part of the whole ordeal. You have to know a little about a lot of stuff to get it all working. Here's where the help services of a good ISP can make a difference. They might have good written instructions or patient, competent technical support people. Before you try connecting, you must have several pieces of information:

- the ISP modem phone number

I believe that what I am suggesting in this article is somewhat radical and on the fringe. I do not know for certain how well it will work out. Think of this as an experimental course – we'll see how the students do. I'd like to hear from readers how things go, and I'll post experiences and lessons on <www.boulderlabs.com> .

- the account username and password
- possibly miscellaneous CHAT scripts
- the IP address of the nameserver
- the IP address that your ISP wants you to use.

PPP has some black magic in it. With luck, you'll get IP connectivity with no problem. Otherwise you'll need to try things, read manuals and FAQs, and get help.

The Connection

Since we are talking cheap here, you probably won't be setting up frame relay or ISDN. For most novices, basic modem speeds of 28.8K or 33.6K will be just fine. But if your ISP has 56K, you have a capable modem (they are now called v.90), and the phone lines between you are capable if it, you might as well have the faster speed. (Note that a large percentage of people cannot use 56K because of the Telco infrastructure.) In my area (US West territory), ADSL is making a big splash. For about \$65 per month, you get your regular voice line and Internet access at speeds of 256K and up. You have some significant up-front costs, but thereafter you've got a great setup.

Word Processing and Other Applications

The big stumbling block for UNIX has been word processing or compatibility with the rest of the word-processing world. I used to think that if you wanted Microsoft application capability under UNIX, you had to run an emulator such as the commercial, expensive SoftWindows. Then you had also to buy the Microsoft applications. I found the whole combination was marginal. Just recently I discovered StarOffice <<http://www.stardivision.com>> that gives you a word processor, a spreadsheet and other applications that have the "look and feel" of Microsoft Word and Excel. (Thank goodness the look and feel lawsuit didn't stand!) These applications not only mimic the expensive commercial ones, they also allow full interchange of documents and material with the commercial ones. So you can receive a Word document, modify it and send it back in Word format. The Word and Excel products are full-featured; if you can do it under Microsoft, you can do it under StarOffice. The PowerPoint like application currently can only read Microsoft's format. They hope by the end of the year to be writing that format too. StarDivision claims: "StarOffice guarantees the direct collaboration with Microsoft Office, without requiring any additional measures." Now the best part; a non-commercial, personal use version is free from the Internet. If you want their CD set and printed manuals, it's only \$40 (this saves a 60MB Internet download.)

Applix <<http://www.applix.com>> offers Applixware, which gives you word processing and spreadsheets, but doesn't try to look exactly like Microsoft. It has filters to deal with the Microsoft formats. Applixware Office Suite 4.3.7 comes with word processor, spreadsheet, presentation graphics, email, and HTML authoring for \$99.95.

OpenLinux from Caldera ships with the free graphical KDE desktop environment, StarOffice 4.0 suite, Netscape Communicator, and the Linux 2.0.35 kernel for \$59.

WordPerfect is available for Linux and the BSDs. It represents a viable option to Word and is the mainstay of the legal profession.

LyX is a front-end for the document processing system LaTeX. It has lots of WYSIWYG features that bring the system closer to its word-processing competitors, while preserving the amazing flexibility of raw LaTeX.

People doing word processing are going to want a printer – probably an inexpensive ink-jet printer rather than an easier-to-deal-with and more expensive Postscript printer.

It used to be hard to drive these couple-hundred-dollar printers from UNIX, but things are rapidly improving. The Ghostscript system can print to a wide variety of low-end printers, including the HP InkJets, the Cannon BubbleJets, Epsoms, and others. See <http://www.cs.wisc.edu/~ghost/prINTER.html> for a list of models.

The list of useful free or inexpensive software that can run on these systems is endless. There are programs to balance checkbooks, track stocks, run image scanners, and manipulate audio. The games doom, chess, backgammon and others are all available. Once your friend handles the basics of surfing and emailing, you can enhance their software suite.

Conclusion

In today's world, the people without Web access and email are at a disadvantage. Plenty of people on tight budgets cannot afford to spend a grand or so on nonessential stuff. Even if you have significant disposable income and could give a computer system, a large circle of friends or too many siblings, in-laws, cousins, etc. could bankrupt you. Alternatively, you might find it challenging and fun to build a useful resource for people you care about. Maybe I've given you some ideas for your holiday list.

Thanks to Tom Poindexter, Rob Kolstad, and Steve Gaede.

torn money and the PGP web of trust

Legitimacy and trust are perhaps the most complicated aspects of PGP (Pretty Good Privacy). The trust model used by PGP assumes that trust starts with bilateral arrangements (key signing) and grows organically to produce a decentralized "web" known as the "Web of Trust." Decentralization is advantageous in that it foregoes the need for any central authority, yet the model as it stands does not scale well in a large, open community. Torn Money has been designed as an authentication service primarily to facilitate the introduction of new users to the Web of Trust and also as a means of enhancing connectivity within the existing web.

Torn Money is a follow-up to the AUUG's PGP Key Signing Service, which, in essence, seeks to maintain and support PGP's decentralized trust model.

Background

PGP is a publicly, and internationally, available privacy program. Essentially, it uses public key cryptographic techniques to allow messages to be exchanged between people across public networks while protecting the privacy of the contents and guaranteeing authenticity of the sender.

by Jeanette McLeod

[<jmcleod@qualcomm.com>](mailto:jmcleod@qualcomm.com)

Jeanette McLeod received a BS from the Northern Territory University (Australia) last year, majoring in mathematics and computer science. She works for QUALCOMM International in Sydney.

and Greg Rose

[<ggr@qualcomm.com>](mailto:ggr@qualcomm.com)

Greg Rose is vice president of USENIX. He runs the Australian development office for QUALCOMM, where his major focus is on cryptography for security and authentication in mobile phones.

The major problem with the Web of Trust is that it has to be big and well connected before it becomes useful . . .

Traditionally, one of the problems with cryptographic systems was “key management.” The key is the secret value that allows information to be encoded and/or decoded. Prior to the development of public key cryptography, the key had to be securely exchanged between parties before they could communicate. Public key systems are designed such that two separate keys are used, one of which can be made public (like a telephone number) while the other is kept secure by the owner (like the telephone itself). In light of this development, it would appear that the problem of key management has been solved.

Unfortunately, this is not the case. Key management is undeniably easier using public key systems, but the question now becomes one of authentication. How do you know, for sure, that the person you are sending a secret message to is really the person he or she claim to be? I could easily get a telephone connected in another name and sit back, waiting for phone calls intended for another person of that name.

One solution to the problem is to introduce the notion of “trusted parties,” that is, people whom you trust to introduce (and therefore authenticate) other parties to you. Using the telephone analogy, you would say secret things on the phone only if someone you trust had given you the telephone number, not if you had just looked it up in the phone book. This is what the PGP documentation refers to as the Web of Trust. Its structure is likened to that of a web because each party involved, trusted by you, can introduce other parties whom you may or may not already know.

Another possible solution is the use of Certification Authorities, thereby enforcing a hierarchical structure on the Web of Trust. What this means is that any public key you acquire must now come with a list of certificates. For example, J. Smith’s public key might come with a certificate from Widgets, Inc., stating that he works for them. In order to establish their authenticity, Widgets, Inc. would also require a certificate from someone asserting that it is a Delaware corporation. To authenticate this, the state of Delaware would need a certificate to verify it was really what it claimed to be, and so on. Eventually, the regression must stop, with a certificate being issued by some omnipresent authority (which, at the moment, is RSA Data Security, Inc.).

Both schemes have flaws. The major problem with the Web of Trust is that it has to be big and well connected before it becomes useful, but the Certification Authority approach assumes the sort of control that is often the reason the parties wanted to communicate privately in the first place.

(The above is intended to be an absolutely minimal explanation of the concepts of public key cryptography and key management. If the concepts are not yet clear, the PGP documentation, which you should eventually read, explains it in more detail.)

Torn Money and the PGP Key Signing Service

In an attempt to expand the Web of Trust, AUUG set up a PGP Key Signing Service in which it acted as an introducer for PGP keys. By virtue of the conferences it held, AUUG was in a position to physically meet with people, verify their identity, and then issue key signatures attesting to their identity. The high public profile of the organization meant that key verification wasn’t difficult, and as the procedures for the key-signing were made public, it was easy to decide what level of trust to place in the authenticity of a key signed by AUUG. However, the service was beginning to introduce a hierarchy into the Web of Trust, with AUUG inadvertently taking on the role of a Certification Authority. The implications of this brought the service to an end, because it was no longer conforming to the PGP trust model. However, the service had one very innovative feature: it did not require people to have their key ready in advance.

“Torn Money” has been designed in the same vein as the Key Signing Service, with its main aim to facilitate PGP key signing. This new service avoids the problems that the Key Signing Service was beginning to encounter while managing to preserve the favorable features – namely, it still allows the verification of those who have not prepared their keys in advance. The inspiration behind Torn Money comes from old spy films, where the possession of a significant half of a torn banknote established a person’s identity. The beauty of such a concept is that it no longer requires an “authority” such as AUUG to oversee the key-signing, the notion of the “torn” banknote means that any two parties can be involved and still effectively identify each other at a later date.

Introduction to Torn Money

PGP signing can occur whenever one interested party meets with another (conferences such as those hosted by AUUG or USENIX are a common forum for such an activity). People wishing to have their keys signed provide acceptable proof of identity together with their PGP fingerprint to the person or persons they wish to have sign their key. Their public key can then later be retrieved for signing from a key server or sent via email, with the supplied fingerprint providing verification of the key’s authenticity. However, this kind of key signing is meaningful only if the interested parties already have PGP keys generated and their fingerprints with them. This is not always the case.

Torn Money sidesteps this issue by providing a way in which interested parties can successfully identify each other at a later date. Conceptually, this means that, upon meeting, interested parties will establish their identities as before and then obtain a “secret.” The possession of this secret is what enables secure future communication. With this in place, those who are unprepared now have the opportunity to create a PGP key at some later time and then communicate the required details to those parties from which they obtained their secret. By revealing the secret they were given, they are able to prove their identity, thus validating their key for signing.

Although this scheme makes it conceptually viable for two unprepared parties to trade details, Torn Money’s primary function is to introduce newcomers to the Web of Trust and enhance connectivity. It is therefore essential that one of the parties involved already belong to the Web of Trust so that his or her signature will act to initiate a newcomer. This person, call this an “expert user,” will effectively become the “owner” of the Torn Money. It is this person’s responsibility to generate and distribute the Torn Money, but he or she is in no way to be considered an “authority.” To such effect, the newcomer is well advised to participate in the Torn Money scheme with as many expert users as possible.

Definition of Torn Money

Torn Money borrows its form from that of a banknote. It is simply a piece of paper containing pairs of related secrets (which function something like a banknote’s serial number). Upon generating a piece of Torn Money, the expert user will be required to enter name, email address, PGP Key ID and fingerprint, and the number of newcomers he or she wishes to sign keys for. This information is required to facilitate future communication between the owner of the Torn Money and the recipients.

The generated piece of Torn Money will contain the owner’s name and PGP fingerprint at the top, as well as a sentence comprising eight four-letter words – the secret. Next, there is a blank table of n rows, where n is the number of newcomer keys the owner elected to sign. This is left blank so that the expert user can note down the name, email address, and identification information (optional) of anyone wishing to participate. Lastly, the remainder of the document is divided into n sections, designed to be “torn” off by the owner and distributed among the participants. Each section contains the

“Torn Money” has been designed in the same vein as the Key Signing Service, with its main aim to facilitate PGP key signing.

Note: The use of Torn Money is in no way restricted to newcomer/expert user pairs. As our overall objective is to increase connectivity within the Web of Trust, established users of PGP who may arrive at a gathering ill-equipped for key signing are also encouraged to use Torn Money.

name of the expert user, his email address, PGP key ID and fingerprint, the Web address of the Torn Money Web site, and eight four-letter words – the participant's secret. (See Appendix for a sample of Torn Money).

After verifying a newcomer's identity, the expert user notes their details in a row in the table and gives him the tear-off section corresponding to his row number. This piece of Torn Money should be kept safe and it is now the only existing link between the expert user's identification information and the new user. For security reasons, it is also vital that no one else has access to the Torn Money, as it contains the new user's secret.

Once the newcomer has generated his own PGP key, he should send email to the expert user(s) for whom he has Torn Money. To be secure, the email should be encrypted using the expert user's PGP key (obtained either from the expert user or a key server and verified with the fingerprint of the newcomer's half of the Torn Money), and signed using the newcomer's PGP keys in order to prove ownership. The content of this mail should comprise the new user's PGP public keys itself and the secret eight four-letter words from the Torn Money.

Upon receiving this message, the expert user must verify the secret received before signing the new key. The new user's secret is derived from a combination of the expert user's secret, her row number in the table on the expert's half of the Torn Money, and the expert's user's name. Thus, the expert user must provide these details exactly to the Torn Money verification program to authenticate the contents of the email message. Once this has been achieved, the expert user can sign the new user's key and return it.

Note: The use of Torn Money is in no way restricted to newcomer/expert user pairs. As our overall objective is to increase connectivity within the Web of Trust, established users of PGP who may arrive at a gathering ill-equipped for key signing are also encouraged to use Torn Money.

Torn Money Generation and Verification

Torn Money can be generated in two ways: either by using the Web interface at the USENIX Web site, or by downloading the source for it and generating it on your own computer. The same option applies to the verification of the procedure – a Web interface is available, and the source for it comes as part of the download for the Torn Money program.

User Support

Once the Torn Money project is complete, full documentation and procedures for use will be made available from the USENIX Web site. At this point in time we envision the users of Torn Money to comprise three distinct groups: new users of PGP seeking connection to the Web of Trust, expert users willing to certify new users, and people wishing to advertise gatherings (e.g., conferences, seminars, etc.) where PGP key-signing or exchange of Torn Money can occur. As such, a series of pages will be dedicated to each group.

Newcomers Instruction Page

In support of new users of PGP and Torn Money, a series of help pages will be made available and the Web addresses included on their piece of Torn Money. These pages will include information on PGP and trust, the function of Torn Money and its usage, links to key servers, and the details of any gatherings at which the exchange of Torn Money can occur.

Expert Users Instruction and Generation Page

A set of pages will also be aimed at established users of PGP who wish to generate their own pieces of Torn Money. These pages will include information on the function of Torn Money and its usage as applicable to an expert user, as well as details on how to generate Torn Money and how to verify responses from recipients. The date and location of any gatherings at which the exchange of Torn Money can occur will be made available, and expert users intending to engage in key-signing (and specifically the distribution of Torn Money) will be given the option to register their attendance at specific functions.

Organizer's Page

As part of the Torn Money key-signing service, support will be given to functions and gatherings at which key signing can occur. This support will be provided through a series of pages on the USENIX Web site that will allow organizers to register their functions as forums for PGP key-signing and the distribution of Torn Money. The time, date, and location of the function will be made publicly available so that expert users may indicate their attendance and hence their willingness to certify new users and newcomers seeking an introduction to the Web of Trust may see when they next have the opportunity to be certified.

All feedback, questions and concerns regarding Torn Money can be directed to Greg Rose and/or Jeanette McLeod. Over time appropriate FAQs will be compiled and posted to the Web site and Torn Money will be revised to better meet user needs.

Concluding Remarks

Successful world wide use of PGP depends on a widespread, well-connected Web of Trust. Torn Money has been designed with this goal in mind. The project is due for completion sometime this fall, and the Web pages discussed in this article will be made available from the USENIX Web site <<http://www.usenix.org>>. Meanwhile, any feedback on the project is welcome, and Torn Money is available for trial usage on request.

Appendix

TORN MONEY FOR Jeanette McLeod

Key Fingerprint: 1B DE 98 8F C8 49 05 4B 82 56 DD DA 67 4E FD B0

Verification: real yawn ntis warm winy peel date rate

<u>No.</u>	<u>Name</u>	<u>Email Address</u>	<u>Id Information (optional)</u>
------------	-------------	----------------------	----------------------------------

0			
---	--	--	--

1			
---	--	--	--

2			
---	--	--	--

3			
---	--	--	--

4			
---	--	--	--

0. Name: Jeanette McLeod
Email: <jmcleod@qualcomm.com>
Key ID: 2C500945
Public Key Fingerprint: 1B DE 98 8F C8 49 05 4B 82 56 DD DA 67 4E FD B0
Verification: quit list burg mesh dare jane afro grad
Help: <<http://www.USENIX.org/tornmoney/newcomer.html>>

1. Name: Jeanette McLeod
Email: <jmcleod@qualcomm.com>
Key ID: 2C500945
Public Key Fingerprint: 1B DE 98 8F C8 49 05 4B 82 56 DD DA 67 4E FD B0
Verification: marc oral tick voss mimi cosh toby pure
Help: <<http://www.USENIX.org/tornmoney/newcomer.html>>

2. Name: Jeanette McLeod
Email: <jmcleod@qualcomm.com>
Key ID: 2C500945
Public Key Fingerprint: 1B DE 98 8F C8 49 05 4B 82 56 DD DA 67 4E FD B0
Verification: boat pike amok fast abbe told held coon
Help: <<http://www.USENIX.org/tornmoney/newcomer.html>>

3. Name: Jeanette McLeod
Email: <jmcleod@qualcomm.com>
Key ID: 2C500945
Public Key Fingerprint: 1B DE 98 8F C8 49 05 4B 82 56 DD DA 67 4E FD B0
Verification: scot cord iris lure doff fuel lazy quad
Help: <<http://www.USENIX.org/tornmoney/newcomer.html>>

4. Name: Jeanette McLeod
Email: <jmcleod@qualcomm.com>
Key ID: 2C500945
Public Key Fingerprint: 1B DE 98 8F C8 49 05 4B 82 56 DD DA 67 4E FD B0
Verification: anne duke lamp mock blat lark gawk lair
Help: <<http://www.USENIX.org/tornmoney/newcomer.html>>

using java

Security Outside the JVM

In a previous *login:* article ("Is Java Secure?" August 1998) I presented some issues related to security inside the JVM.

The focus of this article is security outside the JVM. Specifically, I will examine the Java Cryptography Architecture (JCA), which is in the package "java.security" of the jdk1.1.* release. I'll examine the design of the JCA and present a sample Message Digest (MD5) example.

The JCA is an interface for application developers who wish to use the cryptographic functionality and for those who wish to provide cryptographic functionality; the latter are called "providers."

The security package is based on public key cryptography in which each party participating in a secret conversation has a public key and a private key. The cryptographic functionality refers to message digests, signing messages, public key certification, and many other functions.

How the JVM and JCA Work Together

Java environments can take advantage of cryptographic functionality by attaching to a class file of an applet, the digital signature of the person who wrote the applet. The browser that downloads the applet can examine the digital signature and enforce some policy regarding the "trustability" of the applet. The JDK1.1 appletviewer and HotJava browser will support signed applets.

Regardless of how Java-enabled environments can make use of JCA, it is clear that applications such as electronic commerce (and other transaction-based applications) can make good use of JCA.

Message Digests

As the name suggests, a message digest is a condensed representation of some data. It is a representation in the sense that it is very difficult (not impossible) to find another message that has the same digest. Message digests are commonly provided in most cryptographic software, and so it makes sense to examine it here.

Many algorithms can be applied to provide message digest capability. Some well-known algorithms are MD2, MD5, SHA (signature hash algorithm). Using the JDK security package, it is possible to choose any of these. Naturally, the choice is made by the user on the basis of the stringent security requirements of the application.

JCA Design

JCA has been designed with both "users" and "providers" in mind. For instance:

Users (who need cryptographic support):

- can think in terms of concepts
- can request a particular functionality and algorithm
- can specify a particular implementation.

So if an MD5 algorithm is available from two different providers, the application developer can specify not only that they wish to use the MD5 algorithm, but also that it should be a specific provider's implementation.

Providers (those who provide cryptographic functionality):

- implement a set of security services



by Prithvi Rao

Prithvi Rao is the founder of Kiwilabs, which specializes in software engineering methodology and Java training. He has worked on the development of the MACH OS and a real-time version of MACH, and he holds two patents resulting from his work on mobile robots.

<prithvi+@kiwilabs.com>

Providers can use the basic classes to provide cryptographic functionality. The same functionality can be provided by various providers, and they can work together.

- can coexist; a particular functionality can be provided by more than one provider.
- can work together.

Providers can use the basic classes to provide cryptographic functionality. The same functionality can be provided by various providers, and they can work together. For example, support for PGP (Pretty Good Privacy) keys from one provider can work with keys of another provider. This is significant when you consider that code reusability might result in using classes that are not using PGP keys from the same provider (yet another example of why understanding interoperability is important in a different context). Another example is the ability to take the MD5 output of one provider and have it compared with MD5 implementation of another provider.

JCA Classes

The classes that are part of the `java.security` package consist of a collection of core classes. These core classes are referred to as engines because they encapsulate the actual mechanism that does the work.

Each class provides a particular functionality. Some example classes are:

```
public abstract class MessageDigest{...}
public abstract class Signature{....}
```

The file `MessageDigest.java` contains the class definition for message digests. Another core class is `Signature` and is used for electronic signing of data. All of these core classes are abstract classes. In other words they encapsulate a concept rather than a specific algorithm.

Core classes must be subclassed for each implementation of a particular algorithm:

```
public class MD5 extends MessageDigest{...}
public class SHA extends MessageDigest{...}
public class SHAWithDSA extends Signature{...}
```

A provider wishing to provide a particular functionality must do so by first extending the core class for that functionality and then providing an implementation. This is why these abstract classes are called “engines.” The subclass `SHA`, for example, will define a method that does the work of SHA, and similarly `SHAWithDSA` will contain a method which does the work of computing the hash using the SHA algorithm and then encrypting the computed hash using the DSA (Digital Signature Algorithm) algorithm.

Users who wish to use these implementations instantiate a class of particular provider:

```
MessageDigest md = MessageDigest.getInstance("MD5");
MessageDigest md = MessageDigest.getInstance("MD5", "NSA");
Signature sig = Signature.getInstance("DSA", "PrettyGoodSecurity");
```

Each of the core classes contains a static method called `getInstance` which searches for and returns an instance of the engine requested. So in the first example, `md` is an instance of MD5 provided by the “default” provider. The second is an instance of MD5 provided by “NSA.” The third is an example in which `sig` is an instance of the DSA algorithm by a provider called “Pretty Good Security.” There is a configuration file in which you specify the availability of providers.

JCA: An MD5 Example

Let’s take a look at an MD5 application. Here we use a provider’s implementation.

```
import java.io.*;
import java.security.*;
```


The second import must be included because it is not loaded automatically.

```
public class MD5Test
{
    public static void main(String[] args)
    {
        byte[] msg, msgSunDigest;
```

Msg is the buffer for the message for which we want to compute the digest. MsgSunDigest is the buffer which will eventually contain the MD5 digest.

```
File f;
FileInputStream fis;
MessageDigest md5Sun;
```

Md5Sun is the instance of the MD5 digest class. Since it is a subclass of MessageDigest this declaration is ok.

```
int c, nbytes, len;
if(args.length < 1)
{
    System.out.println("Usage: java <file to digest>");
    System.exit(1);
}
try
{
```

Have to put this within a try because many of the methods below throw exceptions.

```
md5Sun = MessageDigest.getInstance("MD5");
// md5Sun = MessageDigest.getInstance("MD5", "SUN");
```

This is where the program creates an instance of the message digest class which implements the MD5 algorithm. The first example asks for any implementation of MD5, and the second requests SUN's implementation, which is actually also the default.

The reason to use getInstance rather than instantiating a class using "new" is that your application might be running on a host in an environment about which you know very little. In order to instantiate a class as "new MD5()" it is necessary to know the package from which it came, and this is not always possible. The danger in requesting a particular provider is that the program will throw an exception if that provider's algorithm is not available. You can always set up exception handlers to try other "well known" providers if the provider of choice throws an UnknownProviderException.

```
f = new File(args[0]);
msg = new byte[len = (int)f.length()];
fis = new FileInputStream(f);
if ((nbytes = fis.read(msg)) != len)
    throw new IOException(len + "unavailable.");
fis.close();
```

Open and read the file containing the message for which we wish to compute the digest.

```
md5Sun.update(msg);
```

Update is an abstract method of the class "security.MessageDigest." It takes a buffer with bytes from the original message and supplies it to the MD5 engine. Here we supply the entire message to update.

```
msgSunDigest = md5Sun.digest();
```

The security package is very intuitive and simple to use. Once again, this is a testimony to Sun's desire to create an easy, extensible, and strong security architecture.

Digest is another abstract method of "security.MessageDigest." This method returns the (MD5) digest of the original message. The MD5 algorithms always produce digests that are 128 bits (16 bytes) long.

```
        System.out.println(md5Sun.toString());  
    }  
}
```

Print out the data.

Running the Program

To run the program try the following steps:

```
unix> javac -g MD5Test  
unix> java MD5Test <name of text file>
```

The output will show the provider's name, the MD5 hash value and the output of the original text file.

Conclusion

The security package is very intuitive and simple to use. Once again, this is a testimony to Sun's desire to create an easy, extensible, and strong security architecture.

Future articles in this series will include an example on how to be a provider and will examine the Java Cryptographic Extensions (JCE).

Readers wishing to get source for the example can send email to the author.

musings

“Make it so.” With these three words, and perhaps only ten minutes before the end of the show, Picard, a now legendary manager (as well as Starfleet captain), sets his crew scurrying into motion. Each crew member has provided suggestions, and Picard has weighed their possible consequences, and made his decision. Well within the assigned time limit, they solve whatever technical problems they had, and peace in the quadrant is again assured.

If only the real world worked so well.

I have been waving my magic wand for years now, to little effect. Perhaps global warming can be blamed on me (all the hot air). Or, perhaps I forgot the magic words: “Make it so.” Maybe it’s the lack of enough venture capital funding? No, just kidding. I have heard from several people what a nightmare quarterly reports can be (unless you are already quite profitable).

And then, someone said to me, “Why don’t you quit waving that thing around, and open your eyes to what already exists?” Well, what a fine idea.

The first thing I noticed was the proliferation of free UNIX. The UNIX operating system was once enslaved, the intellectual property of a very large networking company, the one with the death star logo. You know the one. A very small (at the time) networking company, with strong roots in UNIX and UUCP, decided it was time to take the bull by the horns, and announced that they had created an unencumbered version of UNIX. In order to do this, they had to replace the parts of the Berkeley UNIX kernel that were owned by the big company. Some programmers with no prior knowledge (in particular, Bill Jolitz) were hired to produce the missing pieces.

Virtual Memory

Memory management was one of the more critical, and arcane, missing pieces. By creating and publishing (in *Dr. Dobbs*) this information, UNIX became free. Well, almost. To the bitter end, the large company maintained that it had protected its intellectual property rights. As proof, they could point to the number of companies that had been forced to change the last four digits of their phone numbers from 8649 (UNIX) to something else. The large company’s lawyers would also send yearly letters out to anyone who had misused the adjective “UNIX.” That is, you could not publish something that read “UNIX,” but could say “the UNIX operating system.”

The argument grew quite heated, even as the lawyers became richer and the small company poorer. Fortunately, the large company was not totally humorless – they stipulated that anyone who had ever seen UNIX source code was “contaminated.” That is, anyone who had worked with the large company’s source code would forever be in thrall and unable to produce code that had not been influenced by the experience. Many people in the UNIX community responded by wearing badges labeling themselves as “mentally contaminated.”

Eventually, it was determined that out of thousands of source files, only four could be said to be in dispute. The large company, AT&T, lost, and the rest of us, and UUNET and BSDI, won. BSDI still sells and supports 4.4 BSD UNIX, and three other organizations support free versions of BSD (OpenBSD <<http://www.openbsd.org>>, FreeBSD <<http://www.freebsd.org>>, and NetBSD <<http://www.netbsd.org>>).

Around about the same time, Linus Torvalds, a student in Finland, began working on task switching and the 80386 protected memory mode, writing in assembly. He was



by Captain Farrow

When he’s not exploring other galaxies, Rik Farrow provides UNIX and Internet security consulting and training. He is the author of *UNIX System Security and System Administrator’s Guide to System V*.

<rik@spirit.com>

The roots of UNIX go back almost 30 years. Although Linux is a “new” operating system, it is also based on UNIX. What about something completely different, starting from scratch? Like BeOS.

inspired by Andrew Tanenbaum’s Minix, a UNIX-like operating system not based on an existing code, and used as a teaching tool. With the rudiments of memory management, and a crude disk device driver, Linux version .01 was released in 1991.

Linux most closely resembles System V UNIX, and it is more popular today than Windows NT. Of course, there are more commercial applications that run under NT, but Linux systems don’t have to be rebooted as often. With a cast of thousands working on improving Linux (Microsoft employs between 200 and 300 programmers for NT), it is a wonder that Linux has not grown to be the size of NT 5 (40 million lines of code). Personally, I doubt that would ever happen to a UNIX-like operating system.

Media Driven

The roots of UNIX go back almost 30 years. Although Linux is a “new” operating system, it is also based on UNIX. What about something completely different, starting from scratch? Like BeOS.

BeOS shares little with UNIX – the Bash shell, a POSIX interface, and about 150 UNIX commands (including Perl). It also has memory protection (as any real operating system should), virtual memory, and preemptive multitasking. Beyond that, it is an entirely new operating system which Be PR describes as the “Media OS.” This is perhaps not so far fetched, as the goal of Be is to support digital media – video, audio, 3D models, and other graphics. To accomplish this, BeOS is based on a microkernel design, using message passing and pervasive multithreading. The multithreading is supposed to aid even in single-processor systems by permitting faster preemption.

Be programmers designed BeOS for multiple-processor systems, using a purely symmetric architecture. In asymmetric multiprocessing, one processor acts as “master” and runs the operating system, while other processors, the “slaves” can only run application code. Symmetric systems permit any processor to run OS code, but there usually is a problem. The kernel contains many “critical sections” code that cannot be interrupted and certainly not executed simultaneously by a second processor. The usual solution is to protect critical sections with locks, mutexes, and semaphores, to guard against multiple access. OS designers have a lot of trouble deciding exactly where to place the locks, and most multiprocessor OSs start out with coarse-grained control – locks that guard large hunks of code. Be claims to have built an OS avoiding this problem.

The filesystem supports files 2^{64} in size, or single devices as large as 18 million terabytes (18 billion gigabytes). While this might seem ridiculous today, you can already buy drives that exceed 10 gigabytes, while the maximum file size supported by a 32-bit integer size is 4 gigabytes. Peeking into the BSDI source, the file offsets are also 64 bits, so no problem there.

BeOS does support other filesystems as well: the Mac OS HFS, DOS Fat 16 and 32, and NFS. Currently, all BeOS files appear to be owned by “baron”; there are hooks for adding either UNIX/POSIX-style owner and group, or NT ACLs, as each file includes extended attributes which can hold anything the programmer desires.

Networked

Only TCP/IP is supported, although BeOS can print to AppleTalk printers. As mentioned, there is native support for NFS, FTP, and Telnet, and both an integrated Web browser and rudimentary server are included.

The ties of its founder, Jean-Louis Gassée, and the VP of engineering, Steve Sakoman, to Apple, where they both worked, are obvious. You probably remember when Apple, Inc. considered using BeOS as a replacement for the venerable MacOS, a spaghetti OS

with roots in the early 1980s. I am not very fond of MacOS, because I have to support my wife and son and their PowerPC. An OS that does not protect memory or offer true multitasking is really lame. MacOS 10 will be based on Next's Mach kernel, an improvement I pray I live to see (there was a nice invited talk about this at the USENIX 1998 conference in New Orleans).

Apple did not choose BeOS, but BeOS's native platform is the PowerPC-based Mac. Last spring, a version of BeOS for Pentium systems appeared, but it still has limited hardware support. I really wanted to load this on a "spare" Intel box, just to see how it compared performance-wise to Windows 95, but BeOS for Intel failed to discover the CD-ROM, and does not support SCSI CD-ROMs for installation, leaving out my other Intel boxes.

Although BeOS protects memory and devices, it does provide direct access to video card memory, something that game and graphic designers want (and need). The development platform is C++, with an object-based API. BeOS also includes Silicon Graphics' OpenGL for 3D graphics. Be claims that the objects provided with the system encourage pervasive multithreading, and that the message passing means that disparate applications can communicate without difficulty.

Because BeOS is new, its API does not suffer from having to support old, complex, and backward-compatible interfaces. I must admit this really attracted me; the Win32 API and Mac API are enormous, and X, while smaller, is not very simple. Gcc is included, as is a basic development environment. The official development tool costs \$129. BeOS itself costs \$100 (although it is on sale for \$70 at the time I write this). Check out <www.be.com> for more details.

Enthusiasm

Be appears to have an exciting new OS, with a small, dedicated group of devotees. I wanted to talk about it because it is an interesting approach, with many, but not all, of the features I have been looking for in a new OS. What's missing? Built-in security is important to me. Broader support for hardware is currently lacking, although I imagine that will come (hard to image a high-performance filesystem that cannot run on SCSI drives on Intel platforms). Still, there is an interesting future here for fledgling hackers who want to write graphics software or like having built-in MIDI capabilities.

BeOS is proprietary – no source code. I am looking for something like BeOS, but more open. Still, the price is right, and BeOS appears to be a very good alternative to Windows (Rich Text Format conversion is already supported, and one developer has plans for full conversion of Word files for their word processor).

It seems like I install one operating system a month, mostly upgrades or on new hardware. Sometimes this is easy, like installing BSDI or Linux, where you answer some questions and walk away, or difficult, like NT, where you must babysit for an hour or so. In all cases, dealing with the video adapter and monitor causes the most problems (except in notebooks, where the PC cards are problematical).

Sometimes I wish I could just say, "Make it so," and some minion would hop to it. But then I would miss all the fun.

Because BeOS is new, its API does not suffer from having to support old, complex, and backward-compatible interfaces. I must admit this really attracted me; the Win32 API and Mac API are enormous, and X, while smaller, is not very simple.

java I/O performance



**by Glen
McCluskey**

Glen McCluskey is a consultant with 15 years of experience and has focused on programming languages since 1988. He specializes in Java and C++ performance, testing, and technical documentation areas.

<glenm@glenmccl.com>

In our survey of Java performance issues, an important area to consider is I/O. Java tends to be somewhat insulated from any underlying operating system, and some types of performance issues, for example disk file fragmentation, are not addressable by a Java environment. But other kinds of issues are familiar.

Two of these are method call overhead and buffering. To see how these issues play out, I will present a series of examples, all of which solve the same problem of counting the number of lines in a text file.

In UNIX/C the lowest-level way to read a character from a file is to use the `read()` system call. The equivalent in Java is the `read()` method of `FileInputStream`:

```
import java.io.*;

public class test1 {
    public static void main(String args[])
    {
        try {
            FileInputStream fis =
                new FileInputStream(args[0]);
            int cnt = 0;
            int ch;
            while ((ch = fis.read()) != -1) {
                if (ch == '\n')
                    cnt++;
            }
            fis.close();
            System.out.println(cnt);
        }
        catch (IOException e) {
            System.err.println(e);
        }
    }
}
```

A `FileInputStream` object represents an input stream of bytes from a file. Note that Java characters are two bytes long, and so this stream of bytes doesn't necessarily represent characters in a one-to-one correspondence.

The program requires around 10 seconds to execute on a 2MB text file, one with around 40,000 lines it, using JDK (Java Development Kit from Sun) 1.1.5.

But this approach is kind of low-level, and maybe we want to try something more elegant, using Java library classes and methods that already know about text lines. So a second approach is to say:

```
import java.io.*;

public class test2 {
    public static void main(String args[])
    {
        try {
            FileInputStream fis =
                new FileInputStream(args[0]);
            DataInputStream dis =
                new DataInputStream(fis);
            int cnt = 0;
            while (dis.readLine() != null)
                cnt++;
            dis.close();
        }
    }
}
```



```

        System.out.println(cnt);
    }
    catch (IOException e) {
        System.err.println(e);
    }
}
}

```

`DataInputStream` is a class built on top of an input byte stream. It knows about various sorts of data types, including text lines.

It turns out that this approach is actually slightly slower, because `readLine()` in `DataInputStream` uses the low-level `read()` illustrated above, with attendant method call overhead. `DataInputStream.readLine()` has been deprecated and should not be used in new code. Beyond the performance problems, there are issues with correctly converting bytes to characters.

A newer approach is to use `BufferedReader`, a class that solves the method call overhead, buffering, and conversion problems:

```

import java.io.*;

public class test3 {
    public static void main(String args[])
    {
        try {
            FileReader fr = new FileReader(args[0]);
            BufferedReader br = new BufferedReader(fr);
            int cnt = 0;
            while (br.readLine() != null)
                cnt++;
            br.close();
            System.out.println(cnt);
        }
        catch (IOException e) {
            System.err.println(e);
        }
    }
}

```

This program executes in 1.1 seconds, about ten times faster than the previous two examples.

Is there any faster way to solve this problem? If you're desperate for speed, you can provide your own buffering, and eliminate the step whereby an accumulated input line is converted to a `String` and returned. This approach looks like:

```

import java.io.*;

public class test4 {
    public static void main(String args[])
    {
        try {
            FileInputStream fis =
                new FileInputStream(args[0]);
            int cnt = 0;
            int len;
            byte buf[] = new byte[1024];
            while ((len = fis.read(buf)) > 0) {
                for (int i = 0; i < len; i++) {
                    if (buf[i] == '\n')
                        cnt++;
                }
            }
        }
    }
}

```



```

    }
}
fis.close();
System.out.println(cnt);
}
catch (IOException e) {
    System.err.println(e);
}
}
}

```

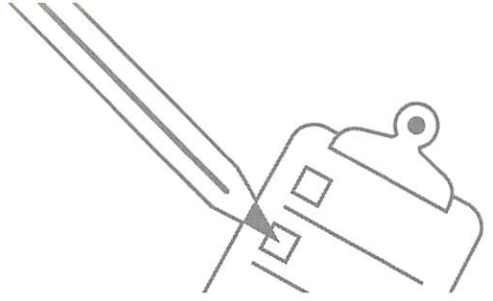
This program executes in 0.3 second, or around 35 times faster than the first two examples. However, manipulating text files in terms of byte streams requires some care, because Java uses the Unicode 16-bit character set, and the notion of a “text file” is somewhat different from what you might be familiar with. A reasonable compromise is to use the technique found in the third example above, a `BufferedReader` object layered on top of a `FileReader`. `FileReader` and `BufferedReader` know about conversion between bytes and characters.

Similar sorts of performance considerations apply to output. For example, a statement like:

```
System.out.println("testing");
```

uses line buffering in support of interactivity, but if you’re willing to disable line buffering, you can improve the output performance considerably.

standards reports



by **Nicholas M. Stoughton**

USENIX Standards Liaison

<nick@usenix.org>

A Programmer's Overview of the POSIX.1d Draft Standard: Additional Realtime Extensions

Karen Gordon <kgordon@vuse.vanderbilt.com>,
Joe Gwinn <gwinn@raytheon.com>,
Jim Oblinger <Oblinger.JT@nuwc.navy.mil>,
and Frank Prindle <prindle@nadc.navy.mil>

Introduction

The POSIX.1d draft standard is the third in a series of realtime POSIX operating system interface standards. The first standard, POSIX.1b-1993, specifies interfaces in nine areas: realtime signals, synchronized I/O, asynchronous I/O, semaphores, memory locking, memory mapped files and shared memory, priority scheduling, high resolution clocks and timers, and message passing. The second standard, POSIX.1c-1995, specifies a number of interfaces that together define a threads extension to the POSIX.1-1990 process model. The standards body responsible for the development and maintenance of all the POSIX standards is the IEEE Portable Applications Standards Committee (PASC). The particular PASC working group that focuses on the realtime POSIX standards is known as the Systems Services Realtime Working Group and is the successor to the fabled POSIX.4 Working Group.

POSIX.1d specifies additional interfaces in support of the realtime goals of predictability and high performance. The interfaces fall into seven areas:

- Process creation via `posix_spawn()`

- Sporadic server execution scheduling policy
- Execution time monitoring of processes and threads
- I/O advisory information
- Timeouts for selected blocking functions
- Device control
- Interrupt control

In the POSIX.1d draft standard, these interfaces are specified as a number of options to POSIX.1, as amended by POSIX.1b and POSIX.1c. Most of the options can be implemented independently of other options. This means that operating system vendors can choose to implement some POSIX.1d interfaces and not others, just as they can choose to implement some POSIX.1b and POSIX.1c interfaces and not others. Likewise, users must decide which optional interfaces they require, and purchase operating systems that implement those required interfaces. It should be noted that some options are dependent on the existence of certain other options. The options and their dependencies are described in the following sections.

As the POSIX.1d draft standard matures, the Working Group will fold subsets of the options specified in the draft standard into existing and/or new application environment profiles. The existing profiles are documented in POSIX.13-1998.

Process Creation Via `posix_spawn()`

The POSIX.1d draft standard introduces interfaces for enabling an application to create a new process – including loading a new process image containing new executable code into the new process's address space in a single step. These interfaces can be used in place of the POSIX.1 process creation interfaces, which entails two steps: first a `fork()` to create a process whose process image is a copy of the parent's process image, and then an `exec()` to overlay the copy of the

The following Report is published in this column:

A Programmer's Overview of the POSIX.1d Draft Standard: Additional Realtime Extensions

Our standards Report Editor, **Nick Stoughton**, welcomes dialogue between this column and you, the readers. Please send any comments you might have to:
<nick@usenix.org>

parent's process-image with a new image, taken from a specified executable file. This two-step process creation mechanism can waste resources; one process image is loaded, and then it is immediately replaced by another process image. More significantly, the two-step mechanism places an undue hardship on systems lacking memory management units (MMUs) or other hardware support for dynamic address translation, because such systems cannot readily implement the `fork()` operation. The problem is that, without dynamic address translation, the addresses in a process image are actual physical addresses. Therefore, the child's process image, if it were indeed an exact copy of the parent's process image, would be competing for the same physical memory as the parent's process image.

The new process creation interfaces, which are available under the Spawn option, are `posix_spawn()` and `posix_spawnnp()`.

Both of these create a new process, whose process image is constructed from a specified executable file referred to as the new process image file. The `posix_spawn()` operation takes a fully defined pathname as an argument; the `posix_spawnnp()` operation builds a pathname from a file argument and the `PATH` environment variable of the calling process. Both operations enable the creator to specify arguments and environment strings for the newly created process. In addition, both operations enable the creator to specify how file descriptors, process group IDs, signal masks, and default signal actions are handled at process creation.

POSIX.5 defines the `Start_Process` Ada language procedure that performs a function nearly identical to that of `posix_spawn()`. While preparing Draft 11 of POSIX.1d, the Working Group considered making `posix_spawn()` a C language binding to `Start_Process`, but realized that doing so would be detrimental to the realtime goals that motivated the development of `posix_spawn()` in the first place. Therefore, the Working Group decided to keep `posix_spawn()` simple and efficient, forgoing some of the power of `Start_Process`. The `Start_Process` procedure allows the caller to specify an arbitrarily long ordered script of file open, close, and duplicate operations to be performed for the new process before the new process image begins execution; the `posix_spawn()` function allows the caller to specify only how each file descriptor in the new process either is mapped to an open file descriptor in the calling process or remains closed. Other than this, `posix_spawn()` provides the same functionality as `Start_Process` and, in fact, `Start_Process` may be directly implemented using `posix_spawn()` in the case where the caller-specified script of file actions is empty.

Sporadic Server Execution Scheduling Policy

The POSIX.1d draft standard supplements the existing POSIX priority scheduling facilities with the sporadic server scheduling

policy, a mechanism for scheduling aperiodic tasks in a primarily periodic environment [Sprunt et al. 89]. The sporadic server scheduling policy was developed as an extension to the rate monotonic scheduling policy. It is based on dynamic changes to the priority of a "sporadic server" set up by an application to service a stream of aperiodic task arrivals; the changes in the priorities are made by the operating system in accordance with parameters supplied by the application and a set of rules spelled out in POSIX.1d. In particular, the priority of the sporadic server is varied between two application-specified levels: a (normal) high foreground priority and a low background priority. The aim of the priority assignment rules is to allow the aperiodic tasks serviced by the sporadic server to execute at a high priority as long as their execution does not pose a threat to the hard deadlines of other tasks.

In short, the sporadic server scheduling policy allows aperiodic tasks to be served at a high priority for a bounded amount of time and at a low background priority at other times. The high-priority service provides the aperiodic tasks with a better average response time than they might otherwise receive. At the same time, the low-priority service that kicks in when they meet their bound on high-priority service prevents them from interfering with hard-deadline periodic tasks, because the aperiodic tasks are then forced to execute in the background and so cannot flood the processor.

The sporadic server scheduling policy is characterized by four parameters, in addition to the normal priority used in POSIX.1b and POSIX.1c:

1. Low priority: the priority level at which the sporadic server executes (i.e., services aperiodic task arrivals) while in the background. While in the foreground, the sporadic server executes at the normal (high) priority.
2. Replenishment period: the length of the time intervals over which the foreground CPU-time usage of the sporadic server is monitored and limited.
3. Initial budget: a bound on the amount of CPU-time that the sporadic server can consume while executing in the foreground during any time interval of length equal to the replenishment period.
4. Maximum number of pending replenishment operations: this value effectively limits the number of aperiodic tasks that can be serviced during any time interval of length equal to the replenishment period. The point of having this limit is to bound the amount of system overhead (storage and CPU time) required to implement the sporadic server scheduling policy.

Under the Process Sporadic Server and Thread Sporadic Server options, the POSIX.1d draft standard adds these four parameters to the process scheduling parameter structure of POSIX.1b and

to the thread scheduling parameter attribute of POSIX.1c. It also amends the following POSIX.1b and POSIX.1c execution scheduling functions to take these parameters into account:

- The POSIX.1b functions for assignment of the scheduling parameters (`sched_setparam()`) and the scheduling policy plus parameters (`sched_setscheduler()`).
- The POSIX.1b functions `sched_getparam()` and `sched_getscheduler()` are implicitly amended. Because of the way the `sched_getparam()` and `sched_getscheduler()` functions are worded in POSIX.1b, it is not necessary to explicitly modify the functions. That is, the descriptions of the functions do not call out other scheduling policies and parameters by name, so it is not necessary to amend the functions to call out the new sporadic server policy and parameters by name. The wording was intentionally designed to make these interfaces extensible in this regard.
- The POSIX.1c functions for assignment and retrieval of scheduling attributes in thread attributes objects for use in thread creation: `pthread_attr_setschedpolicy()`, `pthread_attr_getschedpolicy()`, `pthread_attr_setschedparam()`, `pthread_attr_getschedparam()`.
- The POSIX.1c functions for dynamic assignment and retrieval of thread scheduling policy plus parameters: `pthread_setschedparam()` and `pthread_getschedparam()`. Note that it is implementation-defined whether an application can dynamically change its scheduling policy to the sporadic server scheduling policy.

Execution Time Monitoring of Processes and Threads

The POSIX.1d draft standard supplements the clock and timer facilities of POSIX.1b through two options: the Process CPU-Time Clocks option and the Thread CPU-Time Clocks option. In particular, it defines two new types of clocks:

- Process CPU-time clocks
- Thread CPU-time clocks

In a POSIX.1b clock or timer function call, the CPU-time clock of the calling process is designated by the symbol `CLOCK_PROCESS_CPUTIME_ID`, and the CPU-time clock of the calling thread is designated by the symbol `CLOCK_THREAD_CPUTIME_ID`.

These clocks can be used to monitor the CPU usage of processes and threads, as well as to establish limits on such usage through the setting of timers. As stated in IEEE POSIX.1d, the mechanism used to measure CPU time is implementation-defined; thus, the resolution and accuracy of the CPU-time clocks is implementation-dependent. The software data structure used for representing time, however, is standardized by POSIX.1b, for the sake of application portability. Specifically, the POSIX.1b data

structure provides for nanosecond resolution, although practical computer clocks are not nearly that good.

The functionality provided by CPU-time clocks can be used during the development and operation of many realtime applications, as in the following examples:

- During development, the CPU-time monitoring capability facilitates the collection of information that is vital to system engineers in analyzing the ability of a realtime application to meet its performance specifications.
- During operation, the capability of setting CPU-time limits can be used to enhance the robustness of an application, by enabling the application to prevent a potentially faulty process from capturing the CPU.
- During operation, the capability of setting CPU-time limits can be used to facilitate the scheduling of certain realtime applications, such as those having iterative components whose results become more precise with each iteration. For example, an iterative component might be allowed to execute at a high priority until it reaches its CPU-time limit, at which point it is considered to have achieved an “acceptable” but imprecise result (i.e., a result – with bounded imprecision – that has been shown through a priori analysis to be acceptable, although not optimal, to the application). Then the iterative component might be allowed to execute at a low priority, improving the precision of its result, up until its deadline. At its deadline, the iterative component provides its (possibly still imprecise) result to the application. The idea behind returning an imprecise result at the deadline is that a timely result of bounded imprecision is better than a precise, but late, result (see [Chung et al. 90] for an overview of how to schedule “imprecise computations”).

The measurement of thread execution time may incur excessive overhead in some systems and some applications. Therefore, a new thread creation attribute is introduced by the POSIX.1d execution-time monitoring facility. This attribute allows/disallows thread access to CPU-time clocks; it is set at thread creation-time and is unchangeable thereafter, making it possible for an operating system to optimize the implementation of a given thread.

The specific functions introduced by the POSIX.1d draft standard include the following:

- Getting the clock ID of the CPU-time clock of a specified process (`clock_getcpuclockid()`).
- Getting the clock ID of the CPU-time clock of a specified thread (`pthread_getcpuclockid()`).
- Setting or getting the value of the CPU-time clock thread creation attribute (`pthread_attr_setcpuclockallow()`, `pthread_attr_getcpuclockallow()`).

I/O Advisory Information

The capability of performing I/O operations with deterministic high performance is crucial in realtime systems. To this end, the POSIX.1d draft standard proposes interfaces for enabling an application to give the operating system advisory information on how the application expects to use specified file and memory space. Notably, the application does not tell the operating system how to manage file and memory access; it just offers “hints” relating to characteristics of the application, which the operating system can take into account in making its resource management decisions. The specific interfaces, available under the Advisory Information option, are as follows:

- Providing advisory information on how the application expects to use a specified range of a specified file (`posix_fadvise()`). The information is conveyed through an advice argument that has a number of standard values.
 - `{POSIX_FADV_NORMAL}` No further special treatment
 - `{POSIX_FADV_SEQUENTIAL}` Expect sequential references
 - `{POSIX_FADV_RANDOM}` Expect random references
 - `{POSIX_FADV_WILLNEED}` Will need the specified range soon
 - `{POSIX_FADV_DONTNEED}` Don't need the specified range anymore
 - `{POSIX_FADV_NOREUSE}` Expect data will not be reused once accessed
- Providing advisory information on how the application expects to use a specified range of memory (`posix_madvise()`). This function is available if the Memory Mapped Files option or the Shared Memory Objects option, in addition to the Advisory Information option, is supported. Like the `posix_fadvise()` function, this function conveys information through an advice argument having a number of standard values. The values are labeled `{POSIX_MADV_NORMAL}`, `{POSIX_MADV_SEQUENTIAL}`, `{POSIX_MADV_RANDOM}`, `{POSIX_MADV_WILLNEED}`, `{POSIX_MADV_DONTNEED}`, and have the same meanings as the values used with `posix_fadvise()`.

Note that there is no `posix_madvise()` advice argument value corresponding to the `posix_fadvise()` advice argument value `{FADV_NOREUSE}`. This is because “reuse” of data is accomplished through explicit application-specified sharing of memory in the case of memory mapped files and shared memory objects, whereas reuse is accomplished through operating system buffering in the case of nonmapped files.

Also in the interest of enabling an operating system implementation and an application to work together to optimize performance, the POSIX.1d draft standard introduces the following

new pathname variables which provide the indicated information on files:

Name: `{ALLOC_SIZE_MIN}`

Description: Minimum number of bytes of storage actually allocated for any portion of a file. For direct (unbuffered) I/O, the number of bytes transferred in an I/O operation should be a multiple of `{ALLOC_SIZE_MIN}`. The file offset should also be a multiple of `{ALLOC_SIZE_MIN}`. Valid increments for file transfer sizes between the `{POSIX_REC_MIN_XFER_SIZE}` and `{POSIX_REC_INCR_XFER_SIZE}` `{POSIX_REC_MAX_XFER_SIZE}` values. Note that `{POSIX_REC_INCR_XFER_SIZE}` should be a multiple of `{ALLOC_SIZE_MIN}`.

Name: `{POSIX_REC_MIN_XFER_SIZE}`

Description: Minimum recommended file transfer size.

Name: `{POSIX_REC_MAX_XFER_SIZE}`

Description: Maximum recommended file transfer size.

Name: `{POSIX_REC_XFER_ALIGN}`

Description: Recommended file transfer buffer alignment.

An application can use the POSIX.1 functions `pathconf()` and `fpathconf()` to determine the current value of each of these variables for any given pathname. The application can then use the values to optimally set up its transfers of data between files and memory.

The POSIX.1d draft standard proposes the following additional interfaces, available under the Advisory Information option, that further assist applications in optimizing their performance:

- Pre-allocating or releasing a specified amount of storage space for a specified file (`posix_fallocate()`, `posix_ffree()`). The file size is not affected by these functions. Thus, space can be pre-allocated beyond the current end of the file. This enables append-mode writes to take advantage of the pre-allocation offered by `posix_fallocate()`.
- Allocating a block of memory of a specified size on a specified alignment (`posix_memalign()`). The block can be freed with the C Standard `free()` function [ISO/IEC C].

Timeouts for Selected Blocking Functions

Prior to the development of the POSIX.1d draft standard, some POSIX.1b and POSIX.1c blocking functions had timed versions (i.e., versions that would block subject to a specified timeout period), while others did not. The timed versions included the following:

- The POSIX.1b function `sigtimedwait()`, a timed alternative to `sigwaitinfo()`.
- The POSIX.1c function `pthread_cond_timedwait()`, a timed alternative to `pthread_cond_wait()`.

- The POSIX.1b function `aio_suspend()` with a non-NULL timeout argument, a timed alternative to `aio_suspend()` with a NULL timeout argument.

The Working Group came to view the lack of timed versions of blocking functions as a serious shortcoming, especially in the context of life-critical or mission-critical embedded systems. In these systems, an application must ensure that unbounded blocking can never result from a service request, since unbounded blocking causes the application to lose control of the system. An application's loss of control is intolerable in realtime systems, for it is the application that is supposed to be directing the system in support of the mission. Therefore, an application must be able to detect and to escape from what it considers to be an unreasonable delay in receiving service, and hence to regain control of the system. Upon regaining control, the application may invoke some system-specific fault diagnosis and fault recovery procedures.

Therefore, all the blocking functions of IEEE Std POSIX.1, POSIX.1b, and POSIX.1c were reviewed. It was decided to be unnecessary to supplement blocking I/O services with timed services, because asynchronous (nonblocking) services had already been added to the standard under the IEEE Std POSIX.1b Asynchronous I/O option. In the end, only the following timed functions were added to POSIX.1 (as amended by IEEE Std POSIX.1b and POSIX.1c), under the Timeouts option:

- The function `sem_timedwait()`, as a timed alternative to the POSIX.1b function `sem_wait()` (POSIX.1d, Section 11.2.6).
- The function `pthread_mutex_timedlock()`, as a timed alternative to the POSIX.1c function `pthread_mutex_lock()` (POSIX.1d, Section 11.3.3). The `pthread_mutex_timedlock()` function can be used only with mutexes whose timeout-allowed attribute is set to `PTHREAD_TIMEOUT_ALLOWED`. The timeout-allowed attribute was introduced under the Timeouts option as a performance-preserving mechanism. An application programmer can set the timeout-allowed attributes of selected mutexes to `PTHREAD_TIMEOUT_DISALLOWED` to avoid the overhead associated with the mutexes being set up to handle timed locks.
- The functions `mq_timedsend()` and `mq_timedreceive()`, as timed alternatives to the POSIX.1b functions `mq_send()` and `mq_receive()` (POSIX.1d, Sections 15.2.4-15.2.5).

Device Control

The POSIX.1d draft standard addresses device control in Annex I. This annex, which is informative only (i.e., not a normative part of the draft standard), suggests standardizing the functionality of the traditional UNIX function `ioctl()` in the form of a new function called `posix_devctl()`, which would be available under a Device Control option. In earlier drafts of POSIX.1d,

the `posix_devctl()` function was specified in normative text in the main body of the document. However, some members of the balloting group opposed the inclusion of a device control function in the draft standard, even as an optional interface, and so, in the interest of consensus building, the technical reviewers decided to move the specification of `posix_devctl()` into an informative annex. This section describes the suggested `posix_devctl()` function, the motivation behind it, and some of the objections to it.

The UNIX function `ioctl()`, although proven in practice to provide essential functionality, was recognized as having some room for improvement in its specification. Thus, in designing the `posix_devctl()` function, the Working Group was driven by two somewhat conflicting goals: (1) maintaining compatibility with current `ioctl()` implementations, and (2) establishing sound definitions of the `posix_devctl()` arguments and return value.

The motivation behind the `posix_devctl()` function lies in the fact that the I/O capabilities of POSIX.1 fall short of meeting the I/O needs of realtime applications, as well as other applications. The problem is that many applications need to interact with I/O devices not contemplated by POSIX.1. The applications can choose to interact with such devices in one of two ways: (1) through device drivers or (2) through application code, directly using the POSIX.1d interrupt control facility. In Draft 11 of the POSIX.1d draft standard, Annex I, "Device Control Considerations," addresses the first approach; Annex J, "Interrupt Control Considerations," addresses the second approach.

POSIX.1 specifies general-purpose I/O functions, including `open()`, `close()`, `read()`, `write()`, and `lseek()`. These I/O functions are designed to capture the functionality of random-access mass-storage devices such as disks. Devices whose functionality is not completely captured by the general-purpose I/O functions are considered to be "special devices." POSIX.1 addresses only one type of "special device," terminal I/O. It defines device-specific functions for terminals that enable an application to specify the number of bits per character, the type of parity, the baud rate, etc., for an asynchronous serial communication port.

Realtime systems typically encompass special devices other than terminals. Some of the special devices are common commercially available devices, while others are unique application-specific devices. For common commercially available devices (e.g., magnetic tape drives and printers), it would be theoretically possible, although not necessarily practical, to define a full set of device-specific I/O functions such as those defined for terminals. However, for unique application-specific devices (e.g., specific actuators, sensors, or other controlled devices), it would be impossible to define a full set of device-specific I/O functions, because new devices are continually being developed for new

applications. The functions needed by these yet-to-be-invented devices cannot be anticipated and thus cannot be defined or standardized.

The `posix_devctl()` function that is suggested in Annex I of the POSIX.1d draft standard does not attempt to standardize individual device-specific functions. Instead, it serves as a “standard” mechanism for transmitting any “nonstandard” (i.e., device-specific) I/O commands to any special devices. The `posix_devctl()` function is, in practice, a general application program interface to the device drivers for “special devices.” That is, in the `posix_devctl()` model of communication between application software and device drivers, application programs funnel all device-specific I/O commands through the `posix_devctl()` interface.

The `posix_devctl()` function provides a layer of standardization that has proven to be useful, as evidenced in the widespread use of the `ioctl()` function. The `posix_devctl()` function benefits two groups:

1. Users of device drivers for special devices. Users are given a uniform model of communication between application software and device drivers. The model isolates device-specific application code into readily recognizable locations (i.e., at `posix_devctl()` function calls). Thus, application portability is improved. For example, application software that interacts with a specific analog-to-digital converter can be “ported” relatively easily to interact with another analog-to-digital converter, if the device drivers for both analog-to-digital converters implement the `posix_devctl()` function.
2. Writers of device drivers for special devices. (As previously noted, these writers may be application developers, as in the case of unique application-specific devices.) The model that the `posix_devctl()` function imposes on communication between application software and device drivers serves as a guide to device driver writers. In this way, the `posix_devctl()` function tends to simplify the development of device drivers. In the same way, the `posix_devctl()` function tends to simplify the porting of device drivers when devices need to be moved to different systems.

Like the common `ioctl()` function, the `posix_devctl()` function has the following arguments: (1) a file descriptor of an open device, (2) a driver-specific command requesting the designated device to perform some action, and (3) a pointer to a buffer whose content is command-dependent and therefore also driver-specific. Data is passed between the device driver and the buffer in a command-dependent direction.

The `posix_devctl()` function has two additional arguments: (1) a byte count of the data to be passed between the device driver and the buffer and (2) a pointer to a word (specifically, a data object of type `int`) of driver-specific device information that

may be returned by the function in addition to the usual success/failure indication. In the interest of compatibility with the `ioctl()` function, a byte count of zero can be used to indicate that the amount of data to be passed between the device driver and the buffer is unspecified. Also in the interest of compatibility, the device information word can be used to report information that, in the case of the `ioctl()` function, would be reported via the function return value.

The fact that the `posix_devctl()` function has driver-specific arguments (i.e., the command, the buffer, and the device information word) is the source of the most serious objections to the function. Some balloters do not see the value of the level of standardization provided by the `posix_devctl()` function. In their minds, a function which is by its very nature so implementation-specific (and, moreover, driver-specific) simply is not a candidate for standardization in POSIX.1d or in any other amendment to POSIX.1. On the other side of the controversy, the proponents of the `posix_devctl()` function point to the ubiquity of the UNIX function `ioctl()` as proof of the usefulness of standardizing device control at the “template” level, where the template is standard and the elements of the template are driver-specific.

Interrupt Control

Annex J of the POSIX.1d draft standard proposes an optional interrupt control facility that would make interrupts visible to the application. This facility represents a departure from traditional UNIX practice but is in keeping with realtime kernel practice.

The interrupt control facility, like the device control facility, was at one time specified in normative text in the main body of POSIX.1d, but was moved to an informative annex in response to opposition from some balloters. The opposition stemmed from the same fundamental issue facing device control: interrupt control is inherently implementation-specific. The remainder of this section describes the proposed interrupt control facility and explains how it could be used to improve application portability.

The interrupt control facility put forward in Annex J offers two primary capabilities: (1) An application can associate user-written interrupt service routines (ISRs) with specified interrupts. (2) An application can request to be notified of the occurrence of a specified interrupt. These interfaces are aimed at enabling “connection of nonstandard interrupt-generating hardware in a standard way” [POSIX.1d, Section J.5.1.1]. Here, “nonstandard hardware” means special devices not supported by the operating system vendor. As noted in Section 8, “Device Control,” special devices are common in realtime systems. Application developers could use the POSIX.1d interrupt control facilities to manage special devices in user-level application code. The alternative would be to manage special devices in full-fledged device dri-

vers, installed in the operating system and executed in kernel mode.

The motivation behind the interrupt control facility is given in the POSIX.1d draft standard [POSIX.1d, Section J.5.2]:

Although interrupt handling isn't entirely portable, there is still profit in standardizing the interrupt control interface. First is the implicit standardization of core functionality. Second is programmer portability. Third is that interrupt handling code can follow the hardware device for which it was written. . . . The resulting modularization and isolation of nonportable code also aids portability.

The specific interfaces that would be available under the Interrupt Control option are as follows:

- Associating a specified user-written ISR, with a specified interrupt, or disassociating the ISR and the interrupt (`posix_intr_associate()`, `posix_intr_disassociate()`). The process of associating a user-written ISR with an interrupt is referred to as “registering” the ISR with the operating system.

The target interrupt is specified through an argument of type `intr_t`, whose value identifies an interrupt in an implementation-defined manner.

When registering the ISR, the application specifies the address and size of a “communication region,” an area of memory through which the ISR and the application can exchange data. The communication region is simply an area in the address space of the registering process that the application chooses to make accessible to the ISR. Upon invocation of the ISR, the operating system passes the address of the communication region to the ISR as its first argument. In this way, the ISR can gain access to a region of the registering process's address space, even if the ISR executes in a context different from that of the registering process.

In Annex J of POSIX.1d, the execution context of ISRs is declared to be implementation-defined. In many cases, the ISR will execute as a thread in the context of the operating system or kernel; in other words, it will execute as a kernel thread. In such cases, the address spaces of the registering thread and the ISR are different. This is why a pointer to a “communication region” must be explicitly passed to the ISR upon invocation.

- Specifying to the operating system whether or not the thread that registered a given ISR should be notified when an interrupt is handled by the ISR. This is accomplished through the ISR return value, which is used as a code for indicating (1) whether or not the interrupt was handled and (2) if it was, whether or not the registering thread should also be notified.

The model envisioned in Annex J of the POSIX.1d draft standard is as follows. Multiple devices are typically mapped onto

the same interrupt. For each device, the user may write a separate ISR. Then the multiple ISRs are registered for the interrupt. The registered ISRs are invoked in last-registered- first-invoked order upon an occurrence of the interrupt. When invoked, an ISR must poll its device to determine whether or not its device was the source of the interrupt. If its device was not the source, then the ISR sets its return code to indicate “not handled” and returns immediately. If its device was the source of the interrupt, then it handles the interrupt and sets the return code to indicate “handled” and “notify” or “do not notify,” and it returns.

- Designating a specified segment of application code as a critical section whose execution must not overlap the execution of a specified ISR (or ISRs). Typically, the protected application code and the protected ISR(s) require mutually exclusive access to shared data, in particular, the communication region (or some part thereof) that is identified at the time of ISR registration.

The “protected application code” is the code that falls between `posix_intr_lock()` and `posix_intr_unlock()` function calls. In other words, a thread signifies its intention to enter a critical section by calling the `posix_intr_lock()` function. In the `posix_intr_lock()` function call, the thread specifies an interrupt as the single argument. The “protected ISRs” are the user-written ISRs registered by the calling thread for the specified interrupt.

The protected ISRs cannot begin executing while the protected application code is executing; likewise, the protected application code cannot begin executing until any protected ISR active at the time of the `posix_intr_lock()` call has completed. The mechanisms used to ensure mutual exclusion between protected application code and protected ISRs are implementation defined. Possible mechanisms include operating system (i.e., kernel) mutexes and hardware disabling of interrupts. In addition, several details of these interfaces, such as whether or not locking a given interrupt for a given thread causes other interrupts (e.g., the same interrupt for other threads, or lower-priority interrupts for the same thread or other threads) to also be locked, are also implementation defined.

- Waiting for notification of an (unspecified) interrupt (`posix_intr_timedwait()`). The duration of the wait can be bounded through specification of a timeout argument.

Status of the POSIX.1d Draft Standard

The POSIX.1d draft standard was first balloted at Draft 8 in December 1993. It was recirculated at Draft 10 in March 1997. Due in part to the time that lapsed between the first ballot and the recirculation, many balloters failed to respond to the recirculation. Faced with a nonresponsive ballot group, the Working

Group decided the only way to make progress was to re-form the ballot group. This step was completed in July 1998.

The POSIX.1d draft standard is being reballoted, with the new ballot group, at this time. The results should be available in October 1998. Then ballot resolution will begin. Since the ballot resolution process may lead to changes in the draft standard, the reader should regard this paper as a snapshot of the draft standard as it stands at Draft 11.

References

- [Chung et al. 90] J.Y. Chung, J.W.S. Liu, and K.J. Lin, "Scheduling Periodic Tasks that Allow Imprecise Results," *IEEE Transactions on Computers* 39, 9, 1156-1174.
- [IEEE POSIX.13] POSIX.13-1998 (to be published). For now, available as IEEE Draft Standard POSIX.13/D9 (September 1997), Draft Standard for Information Technology-Standardized Application Environment Profile-POSIX Realtime Application Support (AEP).
- [POSIX.1d] IEEE Draft Standard POSIX.1d/D11 (May 1998), Draft Standard for Information Technology-Portable Operating System Interface (POSIX)-Part 1: System Application Program Interface (API)-Amendment d: Additional Realtime Extensions [C Language].
- [IEEE POSIX.5] POSIX.5-1992, Information Technology-POSIX Ada Language Interfaces-Part 1: Binding for System Application Program Interface (API).
- [ISO/IEC 9945-1] International Standard ISO/IEC 9945-1:1996, Information Technology-Portable Operating System Interface (POSIX)-Part 1: System Application Program Interface (API) [C Language]. Includes IEEE Std POSIX.1-1990, POSIX.1b-1993, POSIX.1c-1995, and POSIX.1i-1995.
- [ISO/IEC C] International Standard ISO/IEC 9899:1990, Information Processing Systems-Programming Languages-C.
- [Liu and Layland 73] C.L. Liu and J.W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment," *Journal of the ACM* 20, 1, 46-61.
- [NGCR 94] U.S. Navy Next Generation Computer Resources (NGCR) Program, "NGCR Operating Systems Standards Group (OSSWG) Advisory: A Programmer's Overview of the IEEE POSIX Realtime Standards" (prepared by K. Gordon, J. Gwinn, J. Oblinger, and F. Prindle), NGCR Document No. OSS-A 001, U.S. Navy Space and Naval Warfare Systems Command, May 1994.
- [Sha and Goodenough 90] L. Sha and J.G. Goodenough, "Real-Time Scheduling Theory and Ada," *IEEE Computer* 23, 4, 53-62.
- [Sha and Sathaye 93] L. Sha and S.S. Sathaye, "A Systematic Approach to Designing Distributed Real-Time Systems," *IEEE Computer* 26, 9, 68-78.
- [Sprunt et al. 89] B. Sprunt, L. Sha, and J. Lehoczky, "Aperiodic Task Scheduling for Hard-Real-Time Systems," *Real-Time Systems* 1, 1, 27-60.
- [TOG 97] The Open Group, Go Solo 2: The Authorized Guide to Version 2 of the Single UNIX Specification, Andrew Josey (editor), Reading, UK, May 1997. Includes "POSIX Realtime" and "POSIX Threads," by K. Gordon, J. Gwinn, J. Oblinger, and F. Prindle, as Chapters 9 and 10.

Note

Due to space limitations, an article on the single European currency by Finnbar P. Murphy will be published in the next regular issue of *login*: in February, 1999. In the meantime, interested readers may find that article at <http://www.usenix.org/publications/login/standards/standards.html>.

the bookworm



by Peter H. Salus

Peter H. Salus is a member of ACM, the Early English Text Society, the Trollope Society, and is a life member of the American Oriental Society. He has held no regular job in the past lustrum. He owns neither a dog nor a cat.

<peter@pedant.com>

I was in San Diego in September for the Tcl/Tk conference, and someone asked me how I could read so many books. I suppressed the urge to tell him that I didn't read the books at all and instead pointed out that a few weeks earlier I had flown Boston to San Jose and back and now Boston-San Diego. In October, it's Boston-LA, and in November, Boston-Amsterdam. At roughly six hours of reading per flight, that's a lot of books consumed inside aluminum cylinders.

One of the (funny) problems that arises is what to do with the "losers." I left one in a plane in August, only to have a flight attendant run after me with it. You can toss magazines and newspapers, but not hard-bound computer books, I guess.

My "top ten" list is at the end of this column; this year it contains several notes concerning items not on the list.

Stringing Along

John Vacca's *Cabling Handbook* is an outstanding piece of work. This is a really comprehensive guide to telecomms and LAN cabling, from Category 5 twisted pair to fiber. Standards, planning, cost-justification, and installation/implementation are all handled in an exemplary fashion. Only one quibble: Vacca does not discuss jacks. The RJ-11 [phone plug], RJ-14 [2-phone modular plug], RJ-22 [handset plug], RJ-45 [8-pin data transmission], RJ-48X [smart jack], etc., are not here. You can't connect those cables without jacks.

AIX

I admit that the last time I used AIX was nearly a decade ago. But Miller has produced a small volume that reads well and appears to combine a tutorial with reference material for those working on RS/6000s. Miller's discussions of migration problems are quite illuminating.

C++

Vandevoorde has produced the perfect adjunct to Stroustrup's third edition of *The C++ Programming Language*: a full explanation and discussion of Stroustrup's exercises, with appropriate cross-references to Stroustrup. Fine work.

Perl

Perl books seem to multiply nearly as fast as Java books did last year. The *Perl Cookbook* tricked me, because it has a bighorn sheep on its cover: after camelidae on the other covers, I didn't expect this. The organization of the *Cookbook* is quite fascinating. It is divided into chapters ("Strings," "Arrays," "File Contents," "Directories," etc.) and each of these contains entries of the form: Problem, Solution, Discussion, and See Also. Really sharp. I found a lot of information that I hadn't been aware of; the organization is solid; the explanations are very fine. Christiansen and Torkington have performed a great service. Now, about that sheep...

Reappearances

The camelidae are still on the cover of the *Perl 5 Pocket Reference*. This second edition is revised for Perl 5.005 and is still the perfect carry-around or desktop item. Vromans has kept up the excellent job he began two years ago.

The second edition of *HTML* brought us up to HTML 3.2; the new third edition takes us to HTML 4.0. Musciano and Kennedy's work has also gained over 50 pages in the past year. Their exposition and explanations are excellent. There is a tearout reference card that's really useful (I replaced my grimy and tattered one from 3.2 two weeks ago).

Books reviewed in this column:

John Vacca

The Cabling Handbook

Upper Saddle River, NJ: Prentice Hall, 1999.
ISBN 0-13-080531-9. Pp. 684.

Bonnie Miller

AIX for UNIX Professionals

Upper Saddle River, NJ: Prentice Hall, 1999.
ISBN 0-13-757246-8. Pp. 184.

David Vandevoorde

C++ Solutions

Reading, MA: Addison Wesley Longman, 1998.
ISBN 0-201-30965-3. Pp. 292.

Tom Christiansen & Nathan Torkington

Perl Cookbook

Sebastopol, CA: O'Reilly & Associates, 1998.
ISBN 1-56592-243-3. Pp. 757.

Johan Vromans

Perl 5 Pocket Reference, 2nd ed.

Sebastopol, CA: O'Reilly & Associates, 1998.
ISBN 1-56592-495-9. Pp. 67.

Chuck Musciano & Bill Kennedy

HTML: The Definitive Guide, 3rd ed.

Sebastopol, CA: O'Reilly & Associates, 1998.
ISBN 1-56592-492-4. Pp. 587.

W. Richard Stevens

UNIX Network Programming, vol. 2, 2nd ed.

Upper Saddle River, NJ: Prentice Hall, 1999.

If you use Perl 5, Vromans is a must; if you use HTML, go to Musciano and Kennedy.

I'm not sure what to write about Rich Stevens's revision of *UNIX Network Programming*. It's really excellent, but I'm certain to say this about his next work, too. Like so many things, it has waxed since its first appearance in 1990. Then it was in one volume of under 800 pages. Volume 2 alone is now 560 pages. And it is concerned with the single topic of interprocess communication. As nearly all programs involve IPCs, this is by no means inappropriate.

Moreover, Stevens really covers the map in terms of different UNIXes. A great example of this is on pp. 460-461, where there are graphs for bandwidth of various types of message passing for Solaris and for Digital UNIX on facing pages.

I wonder what Stevens will turn to next. Perhaps a revision of the TCP/IP volumes for IPv6?

Each year the publishers make it harder for me to put together a list of "best books." They do this by publishing more and more: but they don't expand my reading speed, nor the constraints of the clock and calendar. In 1995 I received 720 books for review; the next year it just topped 1000; last year it was nearly 1300. I'm certain that 1998 will have brought yet more. I read about 100 of these a year. I actually write something about two-thirds of them. The ones below are good books on a variety of topics (one publisher sent me the nth edition of a calculus textbook; I resisted reading it).

The Bookworm's Top Ten for 1998 (not ranked in order)

1. Donald A. Knuth, *The Art of Computer Programming*, Vols. 1, 2, 3. (Addison Wesley Longman)
2. P. Ferguson & G. Huston, *Quality of Service* (Wiley)
3. Berry Kercheval, *TCP/IP over ATM* (Prentice-Hall)

4. Richard Stevens, *UNIX Network Programming*, vol. 2 (Prentice-Hall)

5. Ralph Griswold, et al., *Graphics Programming in Icon* (Peer-to-Peer)

6. B. Schneier & D. Banisar, eds., *Electronic Privacy Papers* (Wiley)

7. Bassam Halabi, *Internet Routing Architectures* (New Riders)

8. Charles Perkins, *Mobile IP* (Addison Wesley Longman)

9. V. Brown & C. Nandor, *McPerl* (Prime Time Freeware)

10. John Blair, *Samba* (SSC)

And two further notes: for real programmers, I think that Harrison & McLennan, *Effective Tcl/Tk Programming* (Addison Wesley Longman) and Johnson & Troan, *Linux Application Development* (Addison Wesley Longman) are truly outstanding; and (in self-interest) you might want to look at the four volumes of P.H. Salus, ed., *Handbook of Programming Languages* (Macmillan).

The ASCII Corporation has published a Japanese translation of Lions' *Commentary and Code*. I am told that Shinichi Iwamoto has done a superb, painstaking job in the translation.

Happy Holidays!

Don Bolinger and Tan Bronson

Applying RCS and SCCS

O'Reilly & Associates. 1995. ISBN 1-56592-117-8. Pp. 501. \$29.95.

Reviewed by Nick Christenson
<npc@jetcafe.org>

Some of the most powerful tools available to an information systems professional are the version control systems. These software packages are designed to help track changes made to line-oriented text documents. Although most commonly used by teams of programmers to manage source code, these systems are useful for far more than just this one application. Unfortunately, version con-

trol systems are also some of the most underutilized tools available to IS folks. Applying RCS and SCCS describes how one can productively use two of the most popular of these tools. The availability of this information should help encourage the tools and methods this book describes.

The book starts out by describing source and project control, and throughout it explains many of the fundamental concepts of configuration management that are necessary for proper maintenance of software development projects. There's a lot more to be said about configuration management than is printed here.

However, there are many good books on this subject, so the authors should be commended for not taking up the reader's attention with an incomplete effort.

For each concept, the authors explain the general theory behind what they are trying to communicate and then elaborate on how this is implemented separately using the two most popular and widely available version control systems: RCS, the Revision Control System, and SCCS, the Source Code Control System. Because of this, the structure of the book flows naturally, and readers do not have to spend time studying the syntax of a system they don't use. Further, even someone who doesn't use either system can efficiently extract considerable information of value from this book. This multi-track writing approach doesn't always work so well, but the authors do a creditable job of pulling it off here.

The subject of the book is fairly narrow, but frankly, I wish more authors would narrow the focus of their work so that they can write with more detail and provide real-world examples to the reader. Because focus, *Applying RCS and SCCS* provides a great deal of specific, good advice and concrete examples that would have had to be omitted in a broader work.

Once the issues in using RCS and SCCS have been dealt with, the authors turn

book reviews

toward the topic of extending these systems. To do this, they introduce their own extension of these tools, which they call TCCS. Although this illustrates their concepts well, they might have been better off using an existing, widely used if less theoretically applicable, extension, like CVS. I doubt this book will persuade many people to adopt TCCS as is, so this section probably isn't as valuable to the book's potential audience as is the rest of it. However, this section can easily be skimmed.

The only other items that may be considered flaws in this book are that the lack of discussion on nonprogramming applications for version control systems, and of a detachable quick reference card. The first is not a big deal, because large software projects are by far the most complex version control environment one is likely to encounter, and it's not too hard for system administrators and document writers to figure out how to use these tools to manage their less complex environments from the information given. As to the second issue, O'Reilly books have recently come with quick reference cards that are of questionable utility. I would actually find cards that summarized the command options for RCS and SCCS useful, so I'm a bit surprised that they weren't included.

The book is well written, and the material is well worth the time it takes to understand it. Version control tools are extremely powerful and useful, and every information systems professional should be intimately familiar with them. It's my opinion that everyone who uses RCS or SCCS, could benefit from this book, and anyone who does not currently use a version control system should read this book and begin using one of these tools immediately. Further, any programmer who uses version control, even if it's not one of the tools covered by this book, would be well served to look over the general sections on version control issues and project management in general.

Applying RCS and SCCS is a solid book on important tools available to the information systems professional. The structure of the book permits efficient learning of those concepts that are most valuable to the specific reader while providing a narrow enough focus and sufficient depth to impart a solid understanding of the theory and practice of document control. If there is a chance that the topic of the book might seem useful, it will almost certainly be well worth reading. I give this topic my strongest recommendation, and this book does not let us down.

Martin Freiss

Protecting Networks with SATAN

O'Reilly & Associates, 1998. ISBN: 1-56592-425-8. Pp. 112. \$19.95.

Reviewed by Nick Christenson

<npc@jetcafe.org>

In 1995, the computer security tool known as SATAN was released, prompting numerous articles in the popular press predicting that it would touch off an avalanche of security break-ins that would devastate the Internet. Of course, as the authors predicted, this catastrophe never occurred, and SATAN took its place as a very useful tool for discovering security vulnerabilities in one's network. In 1997, Martin Freiss wrote *Protecting Networks with SATAN* as a guide to using this powerful software tool. It was originally published in German and now has been translated into English by Robert Bach.

The book begins with a brief description of network security and then describes how to obtain, build, and install SATAN. Next, Freiss provides information on how to conduct network scans with SATAN and describes what SATAN can test for and how it works. The book also provides information on writing one's own SATAN modules, suggested SATAN countermeasures, and concludes with some general security advice in a chapter entitled "Beyond SATAN."

The writing isn't top flight, but the explanations are clear and the prose is no worse than I'd expect from a competently translated text. The information it provides appears correct to me, although I found the sections on general network security to be pretty sparse. Since they're only provided as background and the reader is referred to excellent sources in the bibliography, this isn't a problem. The book is short, but this is no disgrace. The author has chosen a narrow topic to focus on, has explained it with sufficient clarity, and has wisely elected not to waste the reader's time with filler material. I applaud the brevity of the book and wish that more technical authors would consider writing this efficiently.

It seems to me that the main reasons why the press felt that SATAN's release could result in such widespread chaos were its ease of use and familiar Web interface. In my opinion, the most remarkable qualities of this software package are its intriguing inference engine, which infers connections between vulnerable computers, and its easy extendibility. While the book does a nice job of explaining these features, SATAN's user-friendliness makes one wonder if the book is really necessary at all. I really didn't find much substantive information here that isn't immediately available in the SATAN documentation. What I did find new was in the chapters on detecting and repelling SATAN attacks and extending and adapting SATAN. However, this isn't quite enough to make me able to recommend this book.

I suppose if one is having problems figuring out the documentation, or expects to have some difficulty in writing a SATAN extension, the book would prove useful. Further, at a price less than \$20, it's not an extravagant expense, but if one is already successfully using SATAN, I really don't expect that this book would be all that necessary or even terribly helpful, as the SATAN documentation does a pretty good job of covering the same ground.

USENIX Member Benefits

As a member of the USENIX Association, you receive the following benefits:

Free subscription to ;login:,

the Association's magazine, published six to eight times a year, featuring technical articles, system administration tips and techniques, practical columns on Perl, Java, and operating systems, book and software reviews, summaries of sessions at USENIX conferences, and reports on various standards activities.

Access to papers

from the USENIX Conferences starting with 1993, via the USENIX Online Library on the World Wide Web <<http://www.usenix.org>>.

Discounts on registration fees

for all USENIX Conferences, as many as ten every year.

Discounts

on the purchase of proceedings and CD-ROMS from USENIX conferences

PGP Key Signing service

available at conferences.

Discounts

10% off BSDI, Inc. products.
<www.bsd.com>.

Discounts

10% off Prime Time Freeware publications and software <www.pff.com>

Discounts

20% off Prentice-Hall books <www.bookpool.com>
20% off O'Reilly & Associates publications
<www.ora.com>
10% off Morgan Kaufmann books (give code :ALOG)
<www.mkp.com>

Savings

10%-20% savings on selected titles from McGraw-Hill <www.books.mcgraw-hill.com>, John Wiley & Sons <www.wiley.com/compbooks>, The Open Group <www.opengroup.org>, and the MIT Press (give code UNIX1) <mitpress.mit.edu>

Special subscription rates

15% off *The Linux Journal* <www.ssc.com>
\$5 off *The Perl Journal*
<orwant.www.media.mit.edu/the_perl_journal>

The right to vote

on matters affecting the Association, its bylaws, election of its directors and officers

Optional membership

in SAGE, the System Administrators Guild

For information regarding membership or benefits, please contact

<office@usenix.org>
Phone: 510 528 8649

USENIX news

USENIX International Programs

International Speakers Program

The USENIX Association has recently inaugurated an International Speakers Program. We will assist sister organizations and our affiliate member societies in locating and sending speakers to conferences worldwide. USENIX speakers have presented papers at our conferences on such topics as operating systems, network security, system administration, and languages. USENIX will

- assist your organization in ascertaining appropriate topics and speakers,
- extend invitations,
- if needed, fund transportation and expenses.

We only require that the presentation be promoted as a "USENIX Presentation." Please contact the Association's Executive Director, Ellie Young at <ellie@usenix.org>.

Your proposal should contain the following information:

- Who is the sponsor of the event?
- Contact information (phone, postal and email addresses, URL for the organization and the event).
- A brief description of the event (or of a previous similar event).
- Topic(s) for which you seek a speaker or name(s) of specific speaker(s) you would like to invite.
- Dates, and the time period the speaker will need to be present at the event.
- How much support you require from USENIX, if any.

International Affiliate Program

The European.SE <<http://www.europen.se>> (the Swedish National Group of EurOpen), with 300 members, is the first group to become a USENIX Affiliate Member. All members of EurOpen.SE enjoy all USENIX individual member benefits, except voting privileges. All members of an Affiliate group share a

common expiration date, and a single payment for renewal be made each year. There are many different models for affiliate membership. Please contact the Association's Executive Director, Ellie Young <ellie@usenix.org>.

Co-Sponsorship of Conferences

USENIX has agreements with similar technical groups to co-sponsor events. These arrangements vary depending on the needs of each organization, but typically involve USENIX assisting with planning, promotion, proceedings, and/or logistics. USENIX is currently co-sponsoring the following events:

- The 1st International System Administration and Networking Conference (SANE '98), November 18-20, 1998, in Maastricht, The Netherlands, with NLUUG and Stichting NLnet.
<<http://www.nluug.nl/events/sane98/>>.
- NordU99 – 1st Nordic EurOpen/USENIX Conference in Stockholm, Sweden. Feb 9-12, 1999. <<http://www.europen.se/NordU99/>>
- The 2nd IEEE Workshop on Mobile Computing Systems and Applications
<<http://www.research.att.com/conf/wmcsa99/>>.
- The Fifth Annual International Conference on Mobile Computing and Networking
<<http://www.acm.org/sigmobile/conf>>.
- Agent Systems & Programming Conference (to be held in May, 1999).

Inquiries about co-sponsorship should be sent via email to Ellie Young
<ellie@usenix.org>.

by Peter H. Salus

Peter H. Salus is the author of *A Quarter Century of UNIX* (1994) and *Casting the Net* (1995). He has known Lou Katz for over 40 years.

<peter@pedant.com>

Twenty Years Ago in USENIX

Note the change in title! *login*: (aka *UNIX NEWS*) did not reappear until 1980. In 1999 I'll talk about the Association and its activities.

The January 1979 meeting was to be held in Santa Monica, CA. Nearly a year earlier, Peter Weiner at Interactive Systems asked one of his staff if she could handle the arrangements. She did. And she has coordinated every USENIX conference, symposium, and workshop since.

That's right. 25-27 January was Judy DesHarnais's debut as czarina of the meetings. Steve Holmgren chaired the meeting. There were about 350 in attendance. (Little did Judy realize that a decade later there would be ten times as many attendees at the San Francisco Hilton.)

The Association was growing, as was the use of UNIX. The end of 1978 saw an "installed" board of directors, the appearance of Version 7, the first BSD tape (containing Pascal and "the ex editor"), the Ritchie-Johnson port to the Interdata 8, the Wollongong port to the Interdata 7, the port to 32V by Charlie Roberts and a group at Holmdel, the publication of uucp, and the first meeting of the NLUUG – and, of course, the publication of *The C Programming Language* and the "blue" *BSTJ*.

It was quite a year.

Judy, my personal thanks for all those conferences, workshops, board meetings, and symposia.

by Rob Kolstad

Rob Kolstad is head coach of the USA Computer Olympiad Team

<kolstad@busenix.org>

USA Bags Gold at IOI

Four representatives of the USA Computing Olympiad attended the world championships in Setubal, Portugal on September 4-11, 1998. The International Olympiad on Informatics (IOI) hosted over 250 competitors from over 65 countries.

Adrian Sox, winner of almost every USA contest this year, earned a gold medal in the competition. Adrian is now a freshman at Carnegie Mellon University.

Matt Craighead, 16-year-old USACO wunderkind, won a silver medal in a contest marred by judging irregularities. Matt is now a freshman at MIT, having closed out his USACO career with the best record of any USA participant ever.

The 1998-1999 season is under way, with four contests scheduled before the summer training camp, and a trip to Turkey next fall. Please refer interested students to <www.usaco.org> for information on joining the mailing list and participating in the contests.

Statement of Ownership, Management, and Circulation, 10/1/98

Title: *login*: Pub. No. 1044-6397. Frequency: Bimonthly. Seven issues published annually. Subscription price \$50 individuals and institutions. Office of publication: USENIX Association, 2560 9th Street, Suite 215, Berkeley, Alameda County, CA 94710. Headquarters of Publication: Same. Publisher: USENIX Association, 2560 9th Street, Suite 215, Berkeley, CA 94710. Editor: Rob Kolstad. Managing Editor: Jane-Ellen Long, both located at office of publication. Owner: USENIX Association. The purpose, function, and nonprofit status of this organization and the exempt status for Federal income tax purposes has not changed during the preceding 12 months.

Extent and nature of circulation	Average no. of copies of each issue in preceding 12 months	Actual no. of copies of single issue published nearest to filing
A. Total no. of copies	10,117	10,920
B. Paid and/or Requested Circulation		
Sales through Dealers	0	0
Mail Subscribers	9,439	10,315
C. Total Paid and/or Requested Circulation	9,439	10,315
D. Free Distribution by Mail	100	135
E. Free Distribution Outside the Mail	408	350
F. Total Free Distribution	508	485
G. Total Distribution	9,947	10,800
H. Copies Not Distributed	170	120
I. Total	10,117	10,920
Percent Paid and/or Requested Circulation	95%	96%

I certify that the statements made by me above are correct and complete.
Ellie Young, Executive Director

3rd Symposium on Operating Systems Design and Implementation OSDI '99

Monday-Thursday, February 22-25, 1999 • New Orleans, Louisiana

Tutorial Program Monday, February 22, 1999

3 tutorials covering:

Building Security (for Developers), Marcus J. Ranum, *Network Flight Recorder*

Windows NT Internals, Jamie Hanrahan, *Kernel Mode Systems*

Deploying and Benchmarking Web Caches, Peter Danzig, *Network Appliance and USC*; Alex Rousskov, *National Laboratory for Applied Network Research (NLNR)*

Register now to guarantee your first choice—seating is limited.

Technical Sessions Tuesday-Thursday, February 23-25, 1999

Tuesday, February 23, 1999

Opening Remarks and Keynote Address:

The Blind Men and The Elephant

Jim Gettys, *Compaq Computer Corporation*



We are the blind men examining the Web elephant. The interactions between bandwidth, latency, network transport protocols, access protocols, interfaces, and content of the Web are not commonly understood. I will describe my view of the elephant.

"And so these men of Indostan, disputed loud and long, each in his own opinion, exceeding stiff and strong, Though each was partly in the right, and all were in the wrong! So, oft in theologic wars, the disputants, I ween, tread on in utter ignorance, of what each other mean, and prate about the elephant, not one of them has seen!" — John Godfrey Saxe

Jim Gettys is a Senior Consultant Engineer for Compaq Computer Corporation's Industry Standards and Consortia Group and is a Visiting Scientist at the W3C at M.I.T. Jim is the chair of the HTTP/NG Protocol Design Working Group (PDG) of W3C. Jim is the editor of the IETF "Hypertext Transport Protocol—HTTP/1.1" document.

With Bob Scheifler, Jim is co-designer of the X Window System. Gettys' designed the X Library and contributed to X Window System core protocol. Via the Internet, Gettys coordinated the efforts of contributors both inside and outside Digital to the development of X Windows System, one of the first major software systems to be built in a distributed, collaborative fashion.

I/O

Session Chair: Sean O'Malley, *Network Appliance*

Automatic I/O Hint Generation through Speculative Execution

Fay Chang, Garth A. Gibson, *Carnegie Mellon University*

IO-Lite: A Unified I/O Buffering and Caching System

Vivek S. Pai, Peter Druschel, Willy Zwaenepoel, *Rice University*

Virtual Log Based File Systems for a Programmable Disk

Randolph Y. Wang, *University of California, Berkeley*; Thomas E. Anderson, *University of Washington*; David A. Patterson, *University of California, Berkeley*

Resource Management

Session Chair: Greg Minshall, *Siara Systems*

Resource Containers: A New Facility for Resource Management in Server Systems

Gaurav Banga, Peter Druschel, *Rice University*; Jeffrey C. Mogul, *Western Research Laboratory, Compaq Computer Corporation*

Defending Against Denial of Service Attacks in Scout

Oliver Spatscheck, *University of Arizona*; Larry L. Peterson, *Princeton University*

Self-Paging in the Nemesis Operating System

Steven Hand, *University of Cambridge Computer Laboratory*

Panel Discussion: VM-based Operating Systems

Moderator: Paul Leach, *Microsoft Corporation*

Participants: TBD

VISIT OUR WEB SITE: <http://www.usenix.org/events/osdi99>

3rd Symposium on Operating Systems Design and Implementation OSDI '99

Monday-Thursday, February 22-25, 1999 • New Orleans, Louisiana

Wednesday, February 24, 1999

Kernels

Session Chair: Rob Pike, *Lucent Technologies*

Tornado: Maximizing Locality and Concurrency in a Shared Memory Multiprocessor Operating System

Michael Stumm, Ben Gamsa, Jonathan Appavoo, *University of Toronto*; Orran Krieger, *IBM TJ Watson Research Center*

Interface and Execution Models in the Fluke Kernel

Bryan Ford, Mike Hibler, Jay Lepreau, Roland McGrath, Patrick Tullmann, *University of Utah*

Fine-Grained Dynamic Instrumentation of Commodity Operating System Kernels

Barton P. Miller, Ariel Tamches, *University of Wisconsin*

Real-Time

Session Chair: Mike Jones, *Microsoft Corporation*

ETI Resource Distributor: Guaranteed Resource Allocation and Scheduling in Multimedia Systems

Miche Baker-Harvey, *Equator Technologies, Inc.*

A Feedback-Driven Proportion Allocator for Real-Rate Scheduling

David C. Steere, Ashvin Goel, Joshua Gruenberg, Dylan McNamee, Calton Pu, and Jonathan Walpole, *Oregon Graduate Institute*

A Comparison of Windows Driver Model Latency Performance on Windows NT 4.0 and Windows 98

Erik Cota-Robles, James P. Held, *Intel Corporation*

Distributed Systems

Session Chair: Tom Anderson, *University of Washington*

Practical Byzantine Fault Tolerance

Miguel Castro, Barbara Liskov, *MIT Laboratory for Computer Science*

The Coign Automatic Distributed Partitioning System

Galen C. Hunt, Michael L. Scott, *Microsoft Research*; Michael L. Scott, *University of Rochester*

Works-in-Progress

Moderator: John Hartman, *University of Arizona*

Thursday, February 25, 1999

Virtual Memory

Session Chair: Kai Li, *Princeton University*

Tapeworm: High-Level Abstractions of Shared Accesses

Peter Keleher, *University of Maryland*

MultiView and Millipage – Fine-Grain Sharing in Page-Based DSMs

Ayal Itzkovitz, Assaf Schuster, *Technion—Israel Institute of Technology*

Optimizing the Idle Task and Other MMU Tricks

Cort Dougan, Victor Yodaiken, *New Mexico Institute of Technology*; Paul Mackerras, *Australian National University*

Filesystems

Session Chair: Bruce Lindsay, *IBM Almaden Research Center*

Logical vs. Physical File System Backup

Norman C. Hutchinson, Stephen Manley, Michael Federwisch, Guy Harris, Dave Hitz, Steven Kleiman, and Sean O'Malley, *Network Appliance*

The Design of a Multicast-Based Distributed File System

Björn Grönvall, Assar Westerlund, and Stephen Pink, *Swedish Institute of Computer Science*

Integrating Content-Based Access Mechanisms with Hierarchical File Systems

Udi Manber, *University of Arizona*; Burra Gopal, *Microsoft Corporation*

VISIT OUR WEB SITE: <http://www.usenix.org/events/osdi99>

General Chair
Harel Kodesh
Microsoft Corporation
harelk@microsoft.com

General Vice-Chair
Victor Bahl
Microsoft Research
bahl@microsoft.com

Technical Program Committee Co-Chairs
Tomasz Imielinski
Rutgers University
imielins@cs.rutgers.edu

Martha Steenstrup
BBN Technologies
msteens@bbn.com

Panels Chair
Roy Want
Xerox PARC
want@parc.xerox.com

Tutorial Co-Chairs
Michele Zorzi
University of California, San Diego
zorzi@uuc.ucsd.edu

Krishna Sivalingam,
Washington State University
krishna@eecs.wsu.edu

Workshops Chair
Rahul Tewari
Teledesic Corporation
rahul@teledesic.com

Local Chair
Randy Granovetter
Microsoft Corporation
randygr@microsoft.com

Registration Chair
Sampath Rangarajan
Lucent Technologies Bell Labs
sampath@research.bell-labs.com

Publicity Co-Chairs
Jason Redi
BBN Technologies
jredi@bbn.com

Jorge Cobb
The University of Texas at Dallas
jcobb@utdallas.edu

Steering Committee Chair
Imrich Chlamtac
The University of Texas at Dallas
chlamtac@utdallas.edu

Treasurer
Aaron Woodman
Microsoft Corporation
aaronw@microsoft.com

Technical Program Committee
B.R. Badrinath (Rutgers)
Rajive Bagrodia (UCLA)
Victor Bahl (Microsoft Research)
Mary Baker (Stanford Univ.)
Hari Balakrishnan (MIT)
Daniel Barbara (George Mason U.)
Ramon Caceres (AT&T Research)
Tracy Camp (Colo. Sch. of Mines)
Andrew Campbell (Columbia)
M. Scott Corson (U. of Maryland)
Nigel Davies (Lancaster Univ.)
Maurizio Decina (Polit. di Milano)
Mike Franklin (Univ. of Maryland)
James Freebersyser (ONR)
J.J. Garcia-Luna-Aceves (UCSC)
Pierre Humblet (Eurecom Inst.)
Ravi Jain (Bellcore)
Dave Johnson (CMU)
Anthony Joseph (UC, Berkeley)
Anupam Joshi (Univ. of Maryland)
Jay Kistler (FORE Systems)
Hank Korth (Lucent)
Thomas LaPorta (Lucent)
Murray Mazer (Curl Corp.)
Charles Perkins (Sun)
Ram Ramanathan (BBN)
Ramesh Rao (UCSD)
Mani Srivastava (UCLA)
C-K. Toh (Georgia Tech.)
Malathi Veeraraghavan (Lucent)
Kimberly Wasserman (UMichigan)
Magda El Zarki (Univ. of Penn.)

Announcement and Call for Papers

MobiCom '99

The Fifth Annual International Conference on Mobile Computing and Networking

August 15 - 19, 1999

Seattle, Washington, USA

<http://www.acm.org/sigmobile/conf>

Sponsored by ACM SIGMOBILE

Technically Co-sponsored by the IEEE Communications Society
In cooperation with ACM SIGCOMM, SIGOPS, SIGMETRICS, and SIGMOD;
the USENIX Association; the IEICE; and the IEE



PAPERS: Technical papers describing previously unpublished, original, completed research, not currently under review by another conference, journal, are solicited on the following topics:

- Applications and computing services supporting mobile users
- Network architectures, protocols or service algorithms to cope with mobility, limited bandwidth, or intermittent connectivity
- Database and data management issues in mobile computing
- Design and analysis of algorithms for mobile environments
- Security, scalability and reliability for mobile/wireless systems
- Performance of mobile/wireless networks and systems
- Integration and interworking of wired and wireless networks
- Influence of lower layers on the design and performance of higher layers
- Mobile network protocols
- Mobile agents
- Push and pull data dissemination systems
- Power management
- Wireless multimedia systems
- Satellite communication
- Location-dependent applications
- Distributed system aspects of mobile systems
- Adaptive applications interfaces for mobile systems
- Architectures of mobile/wireless networks and systems
- Traffic integration for mobile applications

All papers will be refereed by the program committee. Accepted papers will be published in the conference proceedings. Papers of particular merit will be selected for publication in the ACM/Baltzer journals Wireless Networks (WINET) and Mobile Networks and Applications (MONET).

(NEW!) SPECIAL PAPERS FOR THE MOBICOM "NEXT CENTURY CHALLENGES" SESSION: MobiCom solicits short papers (8 pages max.) that challenge the mobile computing community with new technologies or visionary applications. Such papers should provide stimulating ideas or visions that may open up exciting avenues of mobile computing research. Papers will be reviewed and should be submitted as regular MobiCom submissions (although they do not have to be as technical as regular submissions). However, the cover letter that accompanies the submission should be clearly marked for "MobiCom Challenges".

HOW TO SUBMIT: All paper submissions will be handled electronically. Authors should E-mail a PostScript version of their full paper to mobicom99@cs.rutgers.edu. This E-mail address will become operational by December 15, 1998. In order to ensure that the PostScript versions of the papers can be printed, authors should be careful that their papers meet the following restrictions:

- PostScript version 2 or later.
- No longer than 15 pages.
- Fits properly on US letter size paper (8.5 X 11 inches).
- Reference only Computer Modern or standard Adobe printer fonts (i.e. Courier, Times, Roman, or Helvetica); other fonts may be used but must be included in the PostScript file.

In addition, authors should separately E-mail the title, author names and full address, and abstract of their paper to the Program Chairs, **Tomasz Imielinski** (imielins@cs.rutgers.edu) and **Martha Steenstrup** (msteens@bbn.com). All submitted papers will be judged based on their quality through double-blind reviewing, where the identities of the authors are withheld from the reviewers. Authors names should not appear in the paper or in the PostScript file.

TUTORIALS: Proposals for tutorials are solicited. Evaluation of proposals will be based on the expertise and experience of the instructors, and on the relevance of the subject matter. Potential instructors are requested to submit a tutorial proposal of at most 5 pages, including a biographical sketch, to the Tutorial Co-Chairs, **Michele Zorzi** (zorzi@uuc.ucsd.edu) and **Krishna Sivalingam** (krishna@eecs.wsu.edu) by January 15, 1999.

PANELS: Panels are solicited that examine innovative, controversial, or otherwise provocative issues of interest. Panel proposals should not exceed 3 pages, including biographical sketches of the panelists. Potential panel organizers should contact the Panels Chair, **Roy Want** (want@parc.xerox.com).

STUDENT PARTICIPATION: Papers with a student as a primary author will be considered for a cash award of US\$500 for the best student paper competition. Student authors should clearly indicate with their submission that they would like to be considered for this award.

IMPORTANT DATES:
Submissions due: January 15, 1999
Notification of acceptance: April 5, 1999
Camera-ready version due: May 15, 1999

FOR MORE INFORMATION: Please contact the Program Co-Chairs: **Tomasz Imielinski**, imielins@cs.rutgers.edu, Tel: +1 732 445-3546, Fax: +1 732 445-1003; or **Martha Steenstrup**, msteens@bbn.com, Tel: +1 617 873-3912, Fax: +1 617 873-6091; or the Local Chair, **Randy Granovetter**, randygr@microsoft.com, Tel: +1 425 703-7446, Fax: +1 425 936-7329. This Call for Papers, as well as other MobiCom'99 information, is available on the Web on the ACM SIGMOBILE Home Page at <http://www.acm.org/sigmobile/>.



**Great ideas fly around here
all the time.
So do a few other things.**

At Sprint Paranet, we take a few things seriously: Technology. Success. Career Development. That's how we've become a leading provider of distributed network management solutions for the Fortune 1000. We'll give you a chance to push the limits, and be rewarded. You'll see your biggest ideas put to use. And since we're vendor-independent, you'll work with a host of technologies. Sound like fun? Then explore the following positions, and watch your ideas — and your career — really take off.

Sr. UNIX Systems Analyst

Position requires 5-10 years' experience administering medium to large environments on multi-platforms (Solaris, HP-UX, and AIX). Must have expert networking knowledge of DNS, NFS, NIS (+), Automounter and TCP/IP. Strong shell scripting development skills (C, C++, Perl, AWK, Bourne, Korn) is a plus. UNIX Security and Disaster Recovery projects experience desired.

UNIX Systems Analyst

Must have experience administering medium to large environments on multi-platforms including Solaris, HP-UX, AIX, NFS, NIS+, DNS, Automounter, TCP/IP and multi-platform integration.

Ready to join Sprint Paranet? Then bring your best ideas with you. Because in addition to an excellent compensation package, we provide an open, innovative environment where creativity flourishes. For consideration, please forward your resume to: **Sprint Paranet, Job Code LOGIN1299; Fax (888) 525-7476; Email: eagles-login@sprint-paranet.com (text format is strongly preferred). Please indicate geographic preference. EOE, m/f/d/v**

www.sprintparanet.com

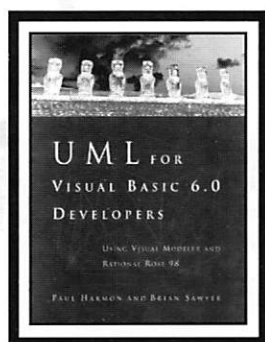


Sprint Paranet
Integrating Hardware, Software,
Networks & People

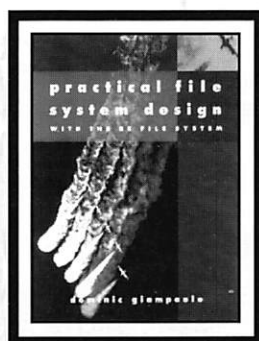


MORGAN KAUFMANN PUBLISHERS

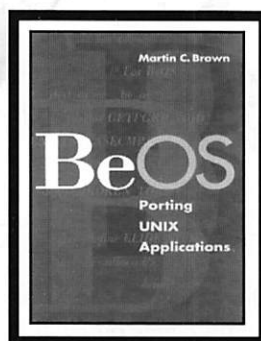
GOT QUESTIONS? WE HAVE ANSWERS!



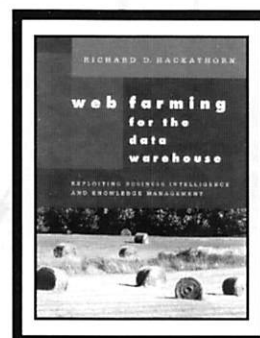
UML for Visual Basic 6.0 Developers
Using Visual Modeler and Rational Rose 98
By Paul Harmon & Brian Sawyer
November 1998; 400 pages; paper;
ISBN 1-55860-545-2; \$39.95



Practical File System Design
with the Be File System
By Dominic Giampaolo
November 1998; 256 pages; paper;
ISBN 1-55860-497-9; \$34.95



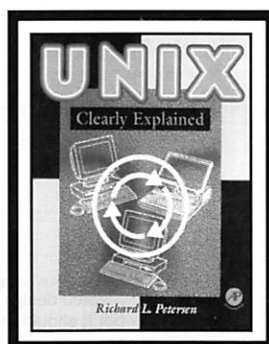
BeOS
Porting Unix Applications
By Martin C. Brown
August 1998; 496 pages; paper;
ISBN 1-55860-532-0; \$44.95



Web Farming
for the Data Warehouse
By Richard Hackathorn
November 1998; 384 pages; paper;
ISBN 1-55860-503-7; \$44.95



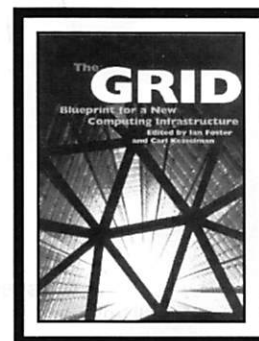
The Jasmine Object Database
Multimedia Applications for the Web
By Setrag Khoshafian, Surapol Dasananda
& Norayr Minassian
August 1998; 432 pages; paper;
ISBN 1-55860-494-4; \$49.95



Unix Clearly Explained
By Richard L. Petersen
October 1998; 709 pages; paper;
ISBN 0-12-552130-8; \$39.95

ALSO FORTHCOMING:

Unix Networking
Clearly Explained
By Richard L. Petersen
February 1999; 500 pages; paper;
ISBN 0-12-552145-6; \$39.95



The Grid
Blueprint for a New
Computing Infrastructure
Edited by Ian Foster & Carl Kesselman
July 1998; 701 pages; cloth;
ISBN 1-55860-475-8; \$62.95

Order from Morgan Kaufmann Publishers
Mail: Harcourt Brace & Co., Attn. Order Fulfillment Dept.,
6277 Sea Harbor Drive, Orlando, FL 32887,
Phone: US/ Canada 800-745-7323, 407-345-3800 (Intl.)
Fax: 800-874-6418, 407-345-4060 (Intl.),
Email: orders@mkp.com **Web:** www.mkp.com Also available at your local bookstores!

10% Discount
to Usenix Members!

Please mention code 49214 when ordering.

TECHNOLOGY AND PRIVACY

The New Landscape

edited by Philip E. Agre
and Marc Rotenberg

"Commerce, bandwidth, equity of access, copyright—these are some of the battles that are taking place in the Internet environment, but the one that will affect all of us is the battle over privacy.

Technology and Privacy is an excellent survey of the different issues on the different fronts. This is important reading for journalists, legislators, activists, and citizens." — Steve Cisler, Advanced Technology Group, Apple Computer

"[These essays are] brilliant and clearly written. They offer the most innovative thinking about this issue in a decade and a half. Agre truly moves the privacy debate ahead...a significant new contribution to the privacy debate, even among professionals who deal with these issues regularly." — Robert Ellis Smith, Publisher, *Privacy Journal*

280 pp., 13 illus. \$15 paper

original in paperback

COORDINATING THE INTERNET

edited by Brian Kahin and James H. Keller

Once entirely built and supported by the U.S. Government, the Internet is now remarkably decentralized and uninstitutionalized. Its future is not tied to any particular organization, but as it grows in scope, bandwidth, functionality, and "internationality," it will require greater coordination. At the moment it is not clear what kind of coordinating mechanisms and institutions will evolve.

Problems discussed in this book include network address allocation and the management of domain names, intellectual property, and finances. Solutions explored range from bilateral and universal agreements between nations to consensual standards negotiated in the open market.

A publication of the Harvard Information Infrastructure Project
500 pp. \$25 paper

iWarp

Anatomy of a Parallel Computing System

Thomas Gross and David R. O'Hallaron

"There is no doubt that iWarp was an important research effort. This work is significant as an archival record of the innovative research undertaken in the iWarp project." —Siddhartha Chatterjee, University of North Carolina at Chapel Hill

The iWarp is an experimental parallel system designed and built jointly by Carnegie Mellon University and Intel Corporation. This book describes the complete iWarp system, from instruction-level parallelism to final parallel application. The authors present a range of issues that must be considered to get a real system into practice, and also provide a start-to-finish history of the project, including what was done right and what was done wrong.

530 pp., 270 illus. \$45

PRIVACY ON THE LINE

The Politics of Wiretapping and Encryption

Whitfield Diffie and Susan Landau

"A remarkable blend of technical expertise, historical analysis, and provocative policy argument. This is an indispensable book for anyone hoping to get to the bottom of the disputes over cryptography, computer security, privacy, and wiretapping that currently divide the law enforcement, civil liberties, and high tech communities."

— Kenneth W. Dam, University of Chicago

352 pp. \$25

SOFTWARE AGENTS

edited by Jeffrey M. Bradshaw

A comprehensive survey of the state of the art in the design and use of intelligent software agents and in the creation of communication ability between agents. The book presents the views of proponents and critics of software agents, describes how agents are used to enhance learning and provide intelligent assistance to users in situations where traditional direct manipulation interfaces alone are insufficient, and discusses agent-to-agent communication and the use of agents to provide intelligent interoperability in distributed systems and the Internet.

450 pp. \$42 paper

THE EVOLUTION OF C++

Language Design in the Marketplace of Ideas

edited by Jim Waldo

This collection of articles traces the history of C++ from its infancy in the Santa Fe workshop, to its proliferation today as the most popular object-oriented language for microcomputers. Waldo notes in his postscript that in the process of evolving, the language has lost a clearly articulated, generally accepted design center, with no common agreement about what it should or should not do in the future.

279 pp. \$30 paper

now in paperback

INTERNET DREAMS

Archetypes, Myths, and Metaphors

Mark Stefik

"Clever juxtapositioning of essays wrapped in the author's insightful commentary paints a telling picture: the Internet is unique, yet the policies that shape its design and use are often influenced by the metaphors that we ascribe to it...*Internet Dreams* is not just a philosophical argument, but a valuable history (and prehistory) of the Net. In fact, no other book that I'm aware of portrays the philosophical development of the Internet with such depth and perspective." — Byte

440 pp., 24 illus. \$15 paper

The MIT Press • Five Cambridge Center • Cambridge, MA 02142-1399 USA

To order by phone, call 800-356-0343 (US & Canada) or (617) 625-8569.
E-mail orders: mitpress-orders@mit.edu • The operator will need this code: **UNIX1**

<http://mitpress.mit.edu>

motd



by Rob Kolstad

Rob Kolstad is a long-time USENIX member, having served as chair of several conferences and workshops, director on the Board, and editor of *login*. He is also head coach of the USA programming team.

<kolstad@usenix.org>

Reading

Do you read a lot? Ever stop to count it up? I can't believe how many written words pass before my eyes.

First of all, there's email. A dozen or two notes here, a dozen or two there – every day – and it adds up. I like little tiny notes, e.g., “please unify field theory; urgent, due at 2:00.” I hate those long rambling ones where you hear about the weather and the new nieces and nephews. You never quite know what the action item is. In the worst case, you're expected to remember the names and birth dates of the new nieces and nephews the next time you see that email writer.

My email includes a few digests and summaries. I get a daily 10,000 word summary of news in the computer industry. That's a lot of words! I also get *Edupage*, which I really like. It's short and to the point.

Maybe you read netnews (USENET). I skim a tiny number of groups, including movie-reviews, some free UNIX discussions, risks-digest, and a security group. The groups I read are fairly reasonable about signal to noise ratio. Takes 1-5 minutes/day; not that many total words to read (I don't read all the notes). In 1977-1980 I recall reading all of netnews every day. I guess those would be “the good old days.”

Then there's paper. I get 1-3 periodicals per day (six days per week) in addition to the usual letters and solicitations. The load of periodicals is staggering. I used to feel obligated to read them; that didn't last long! Now I look at the cover, skim the table of contents, read (or skim) those articles of interest and toss the mag. Sometimes this process takes less than 20 seconds.

Except for *EE Times* (a weekly of 60-100 pages or so in large tabloid format). I love *EE Times*. They talk about hardware, of course. I don't know much about hardware, so it's always fascinating for me. I love hearing about the new Silicon-Germanium devices (4x faster than silicon with reduced power as well). I love hearing about the new discoveries in physics and then seeing them translated into actual devices or products five years later. It's keen. I look at every headline in *EE Times*.

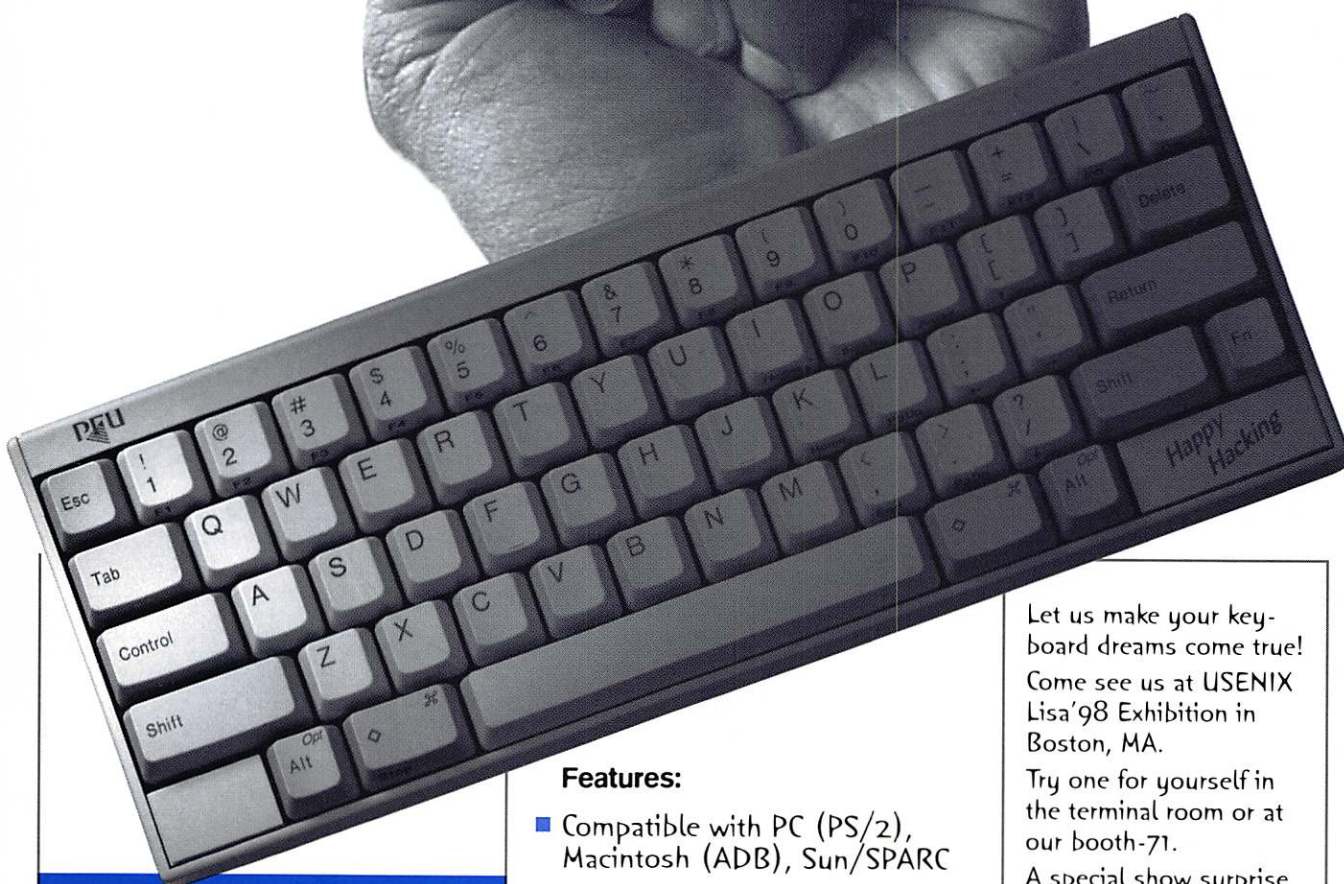
I also run some mailing lists (800 members with three notes/week; and two more (15,000 and 40,000) of one to two per month. It's interesting to hear from everyone – there's a very diverse set of styles and needs.

I also edit newsletters, including this one. Lots of reading there. I edit occasional articles for people; books, too.

I guess it boils down to the idea that I spend most of my day sucking in information and then trying to keep it somewhere. The late Dr. Dan Slotnick (University of Illinois professor) lamented this particular problem by bemoaning the loss of French verbs as he filed more information. It's worse for me; I never took French!

Nevertheless, it's neat to read and process so much information and even share it occasionally. There are so many sources of fairly good data and information that I'm starting to believe that there might even be a bright future for this Internet thing.

Hack It With This!



Happy Hacking Keyboard

The Ultimate Personal Keyboard

Features:

- Compatible with PC (PS/2), Macintosh (ADB), Sun/SPARC
- CTRL key is located where it should be
- No "Windows" key
- No designated Function keys (can be generated with "Fn" key)
- A compact, custom styled, hackers dream come true

Let us make your keyboard dreams come true! Come see us at USENIX Lisa'98 Exhibition in Boston, MA.

Try one for yourself in the terminal room or at our booth-71.

A special show surprise will be given to the first 150 customers.

If you can't wait to know more about the keyboard, contact us at 408-573-7700 or visit our web product page.

<http://www.pfuca.com/products/hhkb>

CONNECT WITH USENIX



MEMBERSHIP AND PUBLICATIONS

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710
Phone: 510 528 8649
FAX: 510 548 5738
Email: <office@usenix.org>



WEB SITE

<http://www.usenix.org>



PGP INFORMATION

All correspondence to:

1998 Operational Key
Key ID: 1024/F6F82613 1997/11/21
USENIX 1998 <pgp@usenix.org>
Key fingerprint = 80 6F B5 48 C2 1A B8 45
48 5F F2 38 E6 41 B0 61



1998 Signing Key

Key ID: 1024/4F75A901 1997/11/21
USENIX 1998 Signature
<<http://www.usenix.org/pgp/pgpsig.html>>
Key fingerprint = 05 FD CF A1 2F 47 00 5C
69 C2 25 E4 66 89 A6 9B



USENIX master key <not-for-email>:

Key ID: 1024/2FEA2EF1 1996/04/08
Key fingerprint = DB A7 50 99 66 E4 8A A9
80 B2 D9 E2 FE DA 00 5E

CONTRIBUTIONS SOLICITED

You are encouraged to contribute articles, book reviews, and announcements to *login*. Send them via email to <login@usenix.org> or through the postal system to the Association office.

Send SAGE material to <tmd@usenix.org>. The Association reserves the right to edit submitted material. Any reproduction of this magazine in its entirety or in part requires the permission of the Association and the author(s).

The closing dates for submissions to the next two issues of *login* are and February 2, 1999, and April 8, 1999.

ADVERTISING ACCEPTED

login offers an exceptional opportunity to reach 9,000 leading technical professionals worldwide.

Contact: Cynthia Deno
cynthia@usenix.org
408 335 9445

USENIX



USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710

POSTMASTER
Send Address Changes to *login*:
2560 Ninth Street, Suite 215
Berkeley, CA 94710

PERIODICALS POSTAGE
PAID
AT BERKELEY, CALIFORNIA
AND ADDITIONAL OFFICES